

Graph Cuts with Interacting Edge Weights – Examples, Approximations, and Algorithms

Stefanie Jegelka Jeff A. Bilmes

September 2, 2014

Abstract

We study an extension of the classical graph cut problem, wherein we replace the modular (sum of edge weights) cost function by a submodular set function defined over graph edges. Special cases of this problem have appeared in different applications in signal processing, machine learning and computer vision. In this paper, we connect these applications via the generic formulation of “cooperative graph cuts”, for which we study complexity, algorithms and connections to polymatroidal network flows. Finally, we compare the proposed algorithms empirically.

1 Introduction

Graphs are frequently used to model problems arising from applications in areas as diverse as operations research, signal processing and machine learning. Graphical representations reveal structure in the problem, and allow reductions to classical problems such as MINIMUM CUT, SHORTEST PATH or MAXIMUM SPANNING TREE, and thereby often lead to efficient and practically useful algorithms.

As a prominent example, the problem of finding a minimum cut (or, equivalently, a maximum flow) has been used as a tool for low-level computer vision [10] (e.g., image segmentation and regularization), probabilistic inference in graphical models [25, 55], and for representing pseudo-boolean functions in computer vision and constraint satisfaction problems [44, 56, 67].

An instance of MINIMUM (s, t) -CUT consists of a (weighted) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ with a set \mathcal{V} of n vertices, a set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ of m directed (or undirected) edges, and a nonnegative weight $w(e)$ for each edge $e \in \mathcal{E}$. The MINIMUM (s, t) -CUT problem is then defined as follows:

Problem 1 (MINIMUM (s, t) -CUT). *Given a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ with terminal nodes $s, t \in \mathcal{V}$, find a cut $C \subseteq \mathcal{E}$ of minimum cost $w(C) = \sum_{e \in \mathcal{E}} w(e)$. A cut is a set of edges whose removal disconnects all paths between s and t .*

A range of very efficient algorithms are known to solve MINIMUM (s, t) -CUT; the reader is referred to [1, 58] for an overview.

In graph cut problems, the cost of any given cut $C \subseteq \mathcal{E}$ is a sum $w(C) = \sum_{e \in C} w(e)$ of edge weights. Such functions are *modular* or, equivalently, *additive* on the edge set \mathcal{E} . Similar modular cost functions commonly occur in other “classical” combinatorial optimization problems such as MINIMUM SPANNING TREE, SHORTEST PATH, or MAXIMUM MATCHING.

Additive edge weights represent several problems quite well, and allow for very efficient exact algorithms. As a result, graph models with modular edge weights are widely applied in practice. Modular edge weights model attractive¹ interactions between at most two variables (nodes) at a time, and preclude the interaction between graph edges. That is, the additive contribution $w(e)$ to the cost $\sum_{e \in C} w(e)$ of a cut C by a given edge $e \in C$ is the same regardless of the cut in which the edge e is considered.

A number of applications, however, are accurately modeled with a more expressive model that allows for a particular form of non-additive interactions between edge weights. We survey some examples and applications in Section 2, some of which demand a model that exceeds the representational power of graph cuts with only modular edge weights (Proposition 1 and Figure 1). These examples serve to motivate our study of a more general model that covers all settings listed in Section 2, and that allows the edge weight function to be a nonnegative, nondecreasing submodular set function, a problem formally defined by Problem 2.

A set function $f : 2^{\mathcal{E}} \rightarrow \mathbb{R}$ defined on subsets of the edge set \mathcal{E} is submodular if it satisfies *diminishing marginal costs*: for all sets $A \subset B \subseteq \mathcal{E}$ and $e \in \mathcal{E} \setminus B$, it holds that $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$.² Submodularity allows one to express concepts such as shared fixed costs and economies of scale. In particular, the cost of an additional edge depends on which other edges are used by the cut. As a result, the cost of a set of edges may be much smaller than the sum of their individual costs. We will therefore say that these edges can *cooperate*. Submodular edge weights can model more complex interactions between edges, or, as a result, interactions between pairs of nodes.

With a submodular cost model on edges, we define the following problem:

Problem 2 (Minimum cooperative cut (MINCOOPCUT)). *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, f)$ with terminal nodes $s, t \in \mathcal{V}$ and a nonnegative, monotone nondecreasing submodular function $f : 2^{\mathcal{E}} \rightarrow \mathbb{R}_+$ defined on subsets of edges, find an (s, t) -cut $C \subseteq \mathcal{E}$ of minimum cost $f(C)$.*

A set function f is *nondecreasing* or *monotone* if $A \subseteq B \subseteq \mathcal{E}$ implies that $f(A) \leq f(B)$. Note that MINCOOPCUT may also be seen as a constrained submodular minimization problem:

$$\text{minimize } f(C) \quad \text{subject to } C \subseteq \mathcal{E} \text{ is an } (s, t)\text{-cut in } \mathcal{G}. \quad (1)$$

As cooperative cuts employ the same graph structures as standard graph cuts, they easily integrate into and extend many of the applications of graph cuts.

¹Attractive interactions between two variables mean that these variables are more likely to take the same label.

²Further basic definitions are given in Section 1.2.

Cooperative graph cuts relate to the recent literature in two aspects. First, a number of models in signal processing, computer vision, security and machine learning are special cases of cooperative cuts, as is discussed in Section 2. Second, recent interest has emerged in the theoretical literature regarding the theoretical implications of extending classical combinatorial problems (such as SHORTEST PATH, MINIMUM SPANNING TREE, or SET COVER) from a sum-of-weights to submodular cost functions [62, 28, 20, 21, 32, 33, 46, 27, 66, 5]. None of these works addresses cuts, however. In this work, we provide lower and upper bounds on the approximation factor of MINCOOPCUT. When discussing relaxations, we also address the flow-cut gap. A celebrated result of [18, 17] states that the dual of the linear programming relaxation of MINIMUM (s, t) -CUT is a MAXIMUM FLOW problem, and that their optimal values coincide. We refer to the ratio between the maximum flow value – the value of the optimal solution to the relaxation of the cut problem and its dual, and the value of the optimal discrete cut solution, as the *flow-cut gap*. For MINIMUM (s, t) -CUT, this ratio is one. We formulate a convex relaxation of MINCOOPCUT whose dual problem is a generalized flow problem, where submodular capacity constraints are placed not only on individual edges, but on arbitrary sets of edges simultaneously. The flow-cut gap for this problem can be on the order of n , the number of nodes. In contrast, the related polymatroidal maximum flow problem [48, 26] (defined in Section 5.1.3) still has a flow-cut gap of one. Polymatroidal flows are equivalent to submodular flows, and have recently gained attention for modeling information flow in wireless networks [38, 37, 13]. Their dual problem is a minimum cut problem where the edge weights are defined by a convolution of local submodular functions [49]. Such convolutions are generally not submodular (see Equation (25)).

1.1 Summary of Main Contributions

In this paper, we survey diverse examples of cooperative cuts in different applications, and provide a detailed theoretical analysis.

- We show an information-theoretic lower bound of $\Omega(\sqrt{n})$ for the general MINCOOPCUT problem.
- We show two complementary families of approximation algorithms. The first relies on substituting the submodular cost function by a tractable approximation. The second family consists of rounding algorithms that build on the relaxation of the problem. Interestingly, both families contain algorithms that use the same partitioning of edges into node incidence sets, but in different ways.
- The above observations are tied to the flow-cut gap of MINCOOPCUT. We provide a lower bound of $n - 1$ on this gap, and relate it to different families of submodular functions.

1.2 Notation and Basic Properties

Throughout this paper, we assume to be given a directed graph³ $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n nodes and m edges, and terminal nodes $s, t \in \mathcal{V}$. The function $f : 2^{\mathcal{E}} \rightarrow \mathbb{R}_+$ is submodular and monotone nondecreasing, where by $2^{\mathcal{E}}$ we denote the power set of \mathcal{E} . We also assume f to be *normalized*, i.e., $f(\emptyset) = 0$. An equivalent alternative definition of submodularity of f is that for all $A, B \subseteq \mathcal{E}$, it must hold that

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B). \quad (2)$$

The function f generalizes commonly used edge weights that are denoted by a weight function $w : \mathcal{E} \rightarrow \mathbb{R}_+$. This common weight function w is *modular*, meaning, it satisfies Equation 2 always with equality. We denote the marginal cost of an element $e \in \mathcal{E}$ with respect to a set $A \subseteq \mathcal{E}$ by $f(e \mid A) \triangleq f(A \cup \{e\}) - f(A)$.

For any node $v \in \mathcal{V}$, let $\delta^+(v) = \{(v, u) \in \mathcal{E}\}$ be the set of edges directed out of v , and $\delta^-(v) = \{(u, v) \in \mathcal{E}\}$ be the set of edges into v . Together, these two directed sets form the (undirected) incidence set $\delta(v) = \delta^+(v) \cup \delta^-(v)$. These edge sets straightforwardly extend to sets of nodes: for a set $S \subseteq \mathcal{V}$ of nodes, $\delta^+(S) = \{(v, u) \in \mathcal{E} : v \in S, u \notin S\}$ is the set of edges leaving S . Without loss of generality, we assume all graphs are simple.

When studying graph cuts, it can be informative to consider the node function $h : 2^{\mathcal{V}} \rightarrow \mathbb{R}_+$, $h(X) \triangleq f(\delta^+(X))$ induced by the edge cost function f for each $X \subseteq \mathcal{V}$. It is well known that if f is a (modular) sum of nonnegative edge weights, then h is submodular [58]. If, however, f is an arbitrary monotone nondecreasing *submodular* function, then this is not necessarily the case, as Figure 1 illustrates. Proposition 1 summarizes some key properties of h :

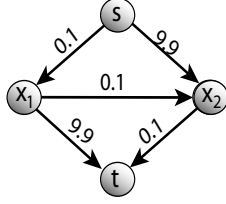
Proposition 1. *Let $h(X) = f(\delta^+(X))$ be the node function induced by a cooperative cut with submodular cost function f .*

1. *The function $h : 2^{\mathcal{V}} \rightarrow \mathbb{R}$ is not always submodular.*
2. *The function h is subadditive, i.e., $h(A) + h(B) \geq h(A \cup B)$ for any $A, B \subseteq \mathcal{V}$.*

The *Lovász extension* $\tilde{f} : [0, 1]^m \rightarrow \mathbb{R}$ of the submodular function f is its lower convex envelope and is defined as follows [49]. Given a vector $x \in [0, 1]^m$, we can uniquely decompose x into its level sets $\{B_j\}_j$ as $x = \sum_j \lambda_j \chi_{B_j}$ where $B_1 \subset B_2 \subset \dots$ are distinct subsets. Here and in the following, $\chi_B \in [0, 1]^m$ is the characteristic vector of the set B , with $\chi_B(e) = 1$ if $e \in B$, and $\chi_B(e) = 0$ otherwise. Then $\tilde{f}(x) = \sum_j \lambda_j f(B_j)$. This construction illustrates that $\tilde{f}(\chi_B) = f(B)$ for any set B . The Lovász extension can be computed by sorting the entries of the argument x in $O(m \log m)$ time.

A *separator* of a submodular function f is a set $S \subseteq \mathcal{E}$ with the property that $f(S) + f(\mathcal{E} \setminus S) = f(\mathcal{E})$, implying that for any $B \subseteq \mathcal{E}$, $f(B) = f(B \cap S) + f(B \setminus S)$.

³Undirected graphs can be reduced to bidirected graphs.



Let $f(A) = \sqrt{\sum_{e \in A} w(e)}$, so
 $h(X) = \sqrt{\sum_{e \in \delta^+(X)} w(e)}$.
 Then h is not submodular:

$$h(\{s, x_1\}) + h(\{s, x_2\}) = \sqrt{19.9} + \sqrt{0.2} < 2\sqrt{10} = h(\{s\}) + h(\{s, x_1, x_2\})$$

Figure 1: The node function implied by a cooperative cut is in general not submodular. The above h violates Inequality (2) for $A = \{s, x_1\}$, $B = \{s, x_2\}$ but satisfies it (strictly) for $A = \{t, x_1\}$, $B = \{t, x_2\}$.

$S) + f(B \cap (\mathcal{E} \setminus S))$. If S is a minimal separator, then we say that f *couples* all edges in S . For the edges within a minimal separator, f is strictly subadditive: $\sum_{e \in S} f(s) > f(S)$. That means, the joint cost of this set of edges is smaller than the sum of their individual costs.

2 Motivation and Special Cases

We begin by surveying special cases of cooperative cuts from a number of applications. The general formulation of (1), however, was first introduced in [32].

Image segmentation. The classical task of segmenting an image into a foreground object and its background is commonly formulated as a *maximum a posteriori* (MAP) inference problem in a Markov Random Field (MRF) or Conditional Random Field (CRF). If the potential functions of the random field are submodular functions (of the node variables), then the MAP solution can be computed efficiently via the minimum cut in an auxiliary graph [25, 8, 45].

While these graph cut models have seen a lot of success in computer vision, they still suffer from known shortcomings. For example, the cut model implicitly penalizes the length of the object boundary, or equivalently the length of a corresponding graph cut around the object. As a result, MAP solutions (minimum cuts) tend to truncate fine parts of an object (such as branches of trees, or animal hair), and to neglect carving out holes (such as the mesh grid of a fan, or written letters on paper). This tendency is aggravated if the image has regions of low contrast, where local information is insufficient to determine correct object boundaries.

A solution to both of these problems is proposed in [32]. It relies on the continuation of “obvious” object boundaries. A prior that replaces the graph cut (pairwise potentials) to a cooperative cut prefers homogeneous, or “con-

gruous”, object boundaries, instead of merely short boundaries. This prior is formally expressed by partitioning the edges in the image grid graph (between neighboring pixels) into groups S_i of similar edges. Similarity is determined by features of the incident nodes. The new, submodular cut weight function grants a discount if few edge groups are used in the cut, i.e., if the object boundary is made up of similar edges, a property often true of objects in real images. The objective is given as

$$f(C) = \sum_{i=1}^k g_i(w(C \cap S_i)), \quad (3)$$

where the g_i are increasing, strictly concave functions, and $w(C) = \sum_{e \in C} w(e)$ is a sum of nonnegative weights. Subsequent work shows exact algorithms for specialized cases [42]. From the viewpoint of graphical models, this function groups the pairwise potentials and introduces a nonlinear dependence between them.

Independently, an alternative submodular cut cost function has been studied to improve image segmentation results:

$$f(C) = \max_{e \in C} w(e). \quad (4)$$

Contrary to the cost function (3), the function (4) couples all edges in the grid graph uniformly, without any similarity constraints. As a result, the cost of *any* long cut is discounted. Sinop and Grady [59] and Allène et al. [2] derive this function as the ℓ_∞ norm of the (gradient) vector of pixel differences; this vector is the edge indicator vector y in the relaxation we define in Section 4. Conversely, the relaxation of the cooperative cut problem leads to new, possibly non-uniform and group-structured regularization terms [31].

Higher-order potentials in computer vision. A range of higher-order potentials from computer vision, i.e., potential functions that introduce dependencies between more than two variables at a time, can be reformulated as cooperative cuts. As an example, P^n Potts functions [40] and robust P^n potentials [41] bias image labelings to assign the same label to larger patches of pixels (of uniform appearance). The potential is low if all nodes in a given patch take the same label, and high if a large fraction deviates from the majority label. These potential functions correspond to a complete graph with a cooperative cut cost function

$$f(C) = g(|C|), \quad (5)$$

for a concave function g [31]. The larger the fraction of deviating nodes, the more edges are cut between labels, leading to a higher penalty. The function g makes this robust by capping the maximum penalty. The distinguishing aspect of P^n models is that, unlike general cooperative cuts, they still lead to a submodular *node cost* function $h(X) = f(\delta(X))$. This submodularity relies on the fact that the graph is complete and that g treats all edges symmetrically.

Regularization and Total Variation. A popular regularization term in signal processing, and in particular for image denoising, has been the Total Variation (TV) and its discretization [57]. The setup commonly includes a pixel variable (say x_j or x_{ij}) corresponding to each pixel or node in the graph \mathcal{G} , and an objective that consists of a loss term and the regularization. The discrete TV for variables x_{ij} corresponding to pixels v_{ij} in an $M \times M$ grid with coordinates i, j is given as

$$\text{TV}_1(x) = \sum_{i,j=1}^M \sqrt{(x_{i+1,j} - x_{ij})^2 + (x_{i,j+1} - x_{ij})^2}. \quad (6)$$

If x is constrained to be a $\{0,1\}$ -valued vector, then this is an instance of cooperative cut — the pixels valued at unity correspond to the selected elements $X \subseteq \mathcal{V}$, and the edge submodular function corresponds to $f(C) = \sum_{ij} \sqrt{C \cap S_{ij}}$ for $C \subseteq \mathcal{E}$ where $S_{ij} = \{(v_{i+1,j}, v_{ij}), (v_{i,j+1}, v_{ij})\} \subseteq \mathcal{E}$ ranges over all relevant neighborhood pairs of edges. Discrete versions of other variants of total variation are also cooperative cuts. Examples include the combinatorial total variation of [15]:

$$\text{TV}_2(x) = \sum_i \sqrt{\sum_{(v_i, v_j) \in \mathcal{E}} \nu_i^2 (x_i - x_j)^2}, \quad (7)$$

and the submodular oscillations in [12], for instance,

$$\begin{aligned} \text{TV}_3(x) &= \sum_{1 \leq i,j \leq M} \max\{x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}\} \\ &\quad - \min\{x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}\} \\ &= \sum_{1 \leq i,j \leq M} \max_{\ell, r \in U_{ij} \times U_{ij}} |x_\ell - x_r|, \end{aligned} \quad (8)$$

$$(9)$$

where for notational convenience we used $U_{ij} = \{(i, j), (i+1, j), (i, j+1), (i+1, j+1)\}$. The latter term (9), like P^n potentials, corresponds to a uniform submodular function on a complete graph, and both (5) and (9) lead to submodular node functions $h(X)$.

Label Cuts. In computer security, *attack graphs* are state graphs modeling the steps of an intrusion. Each transition edge is labeled by an atomic action a , and blocking an action a blocks the set of all associated edges $S_a \subseteq \mathcal{E}$ that carry label a . To prevent an intrusion, one must separate the initial state s from the goal state t by blocking (cutting) appropriate edges. The cost of cutting a set of edges is the cost of blocking the associated actions (labels), and paying for one action a accounts for all edges in S_a . If each action has a cost $c(a)$, then a *minimum label cut* that minimizes the submodular cost function

$$f(C) = \sum_a c(a) \min\{1, |C \cap S_a|\} \quad (10)$$

indicates the lowest-cost prevention of an intrusion [36].

Sparse separators of Dynamic Bayesian Networks. A graphical model $G = (V, E)$ defines a family of probability distributions. It has a node v_i for each random variable x_i , and any represented distribution $p(x)$ must factor with respect to the edges of the graph as $p(x) \propto \prod_{(v_i, v_j) \in E} \psi_{ij}(x_i, x_j)$. A *dynamic graphical model* (DGM) [6] consists of three template parts: a prologue $G^p = (V^p, E^p)$, a chunk $G^c = (V^c, E^c)$ and an epilogue $G^e = (V^e, E^e)$. Given a length τ , an *unrolling* of the template is a model that begins with G^p on the left, followed by $\tau + 1$ repetitions of the “chunk” part G^c and ending in the epilogue G^e .

To perform inference efficiently, a periodic section of the partially unrolled model is identified on which an effective inference strategy (e.g., a graph triangulation, an elimination order, or an approximation method) is developed and then repeatedly used for the complete duration of the model unrolled to any length. This periodic section has boundaries corresponding to separators in the original model [6] which are called the interface separators. Importantly, the efficiency of any inference algorithm derived within the periodic section depends critically on properties of the interface, since the variables within must become a clique.

In general, the computational cost of inference is lower bounded, and the memory cost of inference is exactly given, by the size of the joint state space of the interface variables. A “small” separator corresponds to a minimum vertex cut in the graphical model, where the cost function measures the size of the joint state space. Vertex cuts can be rephrased as standard edge cuts. Often, a modular cost function suffices for good results. Sometimes, however, a more general cost function is needed: Bilmes and Bartels [7], for example, our original motivation for studying MINCOOPCUT, demonstrates that it can be beneficial to use a state space function that considers any deterministic relationships between variables.

An example of a function that respects determinisms is the following. In a Bayesian network that has determinism, let D be the subset of fully deterministic nodes. That means any $x_i \in D$ is a deterministic function of the variables corresponding to its parent nodes $\text{par}(i)$ meaning $p(x_i | x_{\text{par}(i)}) = \mathbf{1}[x_i = g(x_{\text{par}(i)})]$ for some deterministic function g . Let \mathcal{D}_i be the state space of variable x_i . Furthermore, given a set A of variables, let $A_D = \{x_i \in A \cap D \mid \text{par}(i) \subseteq A\}$ be its subset of fully determined variables. If the state space of a deterministic variable is not restricted by fixing a subset of its parents, then the function measuring the state space of a set of variables A is $f(A) = \prod_{x_i \in A \setminus A_D} |\mathcal{D}_i|$. The logarithm of this function is a submodular function, and therefore the problem of finding a good separator is a cooperative cut problem. In fact, this function is a lower bound on the computational complexity of inference, and corresponds exactly to the memory complexity since memory need be retained only at the boundaries between repeated sections in a DGM.

More generally, a similar slicing mechanism applies for partitioning a graph for use on a parallel computer — we may seek separators that requires little information to be transferred from one processor to another. A reasonable proxy for such “compressibility” might be the entropy of a set of random variables, a

well-known submodular function. The resulting optimization problem of finding a minimum-entropy separator is again a cooperative cut.

Robust optimization. Assume we are given a graph where the weight of each edge $e \in \mathcal{E}$ is noisy and distributed as $\mathcal{N}(\mu(e), \sigma^2(e))$ for nonnegative mean weights $\mu(e)$. The noise on different edges is independent, and the cost of a cut is the sum of edge weights of an unknown draw from that distribution. In such a case, we might want to not only minimize the expected cost, but also take the variance into consideration. This is the aim in mean-risk minimization (which is equivalent to a probability tail model or value-at-risk model), where we aim to minimize

$$f(C) = \sum_{e \in C} \mu(e) + \lambda \sqrt{\sum_{e \in C} \sigma^2(e)}. \quad (11)$$

This too is a cooperative cut; in fact, this special case admits an FPTAS [53].

Approximate submodular minimization. Graph cuts have been useful optimization tools but cannot represent any arbitrary set function, not even all submodular functions [67]. But, using a decomposition theorem by Cunningham [16], *any* submodular function can be phrased as a cooperative graph cut. As a result, any fast algorithm that computes an approximate minimum cooperative cut can be used for (faster) approximate minimization of certain submodular functions [34].

3 Complexity and Lower Bounds

In this section, we address the hardness of the general MINCOOPCUT problem. Assuming that the cost function is given as an oracle, we show a lower bound of $\Omega(\sqrt{n})$ on the approximation factor. In addition, we include a proof of NP-hardness. NP-hardness holds even if the cost function is completely known and polynomially computable and representable.

Our results complement known lower bounds for related combinatorial problems having submodular cost functions. Table 1 provides an overview of known results from the literature. In addition, Zhang et al. [66] show a lower bound for the special case of MINIMUM LABEL CUT via a reduction from MINIMUM LABEL COVER. Their lower bound is $2^{(\log \bar{m})^{1-(\log \log \bar{m})^{-c}}}$ for $c < 0.5$, where \bar{m} is the input length of the instance. Their proof is based on the PCP theorem. In contrast, the proof of the lower bound in Theorem 1 is information-theoretic.

Theorem 1. *No polynomial-time algorithm can solve MINCOOPCUT with an approximation factor of $o(\sqrt{|\mathcal{V}|/\log |\mathcal{V}|})$.*

The proof relies on constructing two submodular cost functions f, h that are almost indistinguishable, except that they have quite differently valued minima.

Problem	Lower Bound	Reference
Set Cover	$\Omega(\ln \mathcal{U})$	[28]
Minimum Spanning Tree	$\Omega(n)$	[20]
Shortest Path	$\Omega(n^{2/3})$	[20]
Perfect Matching	$\Omega(n)$	[20]
Minimum Cut	$\Omega(\sqrt{n})$	Theorem 1

Table 1: Hardness results for combinatorial problems with submodular costs, where n is the number of nodes, and \mathcal{U} the universe to cover. These results assume oracle access to the cost function.

In fact, with high probability they cannot be distinguished with a polynomial number of function queries. If the optima of h and f differ by a factor larger than α , then any solution for f within a factor α of the optimum would be enough evidence to discriminate between f and h . As a result, a polynomial-time algorithm that guarantees an approximation factor α would lead to a contradiction. The proof technique follows along the lines of the proofs in [22, 23, 62].

One of the functions, f , depends on a hidden random set $R \subseteq E$ that will be its optimal cut. We will use the following lemma that assumes f to depend on a random set R .

Lemma 1 ([62], Lemma 2.1). *If for any fixed set $Q \subseteq \mathcal{E}$, chosen without knowledge of R , the probability of $f(Q) \neq h(Q)$ over the random choice of R is $m^{-\omega(1)}$, then any algorithm that makes a polynomial number of oracle queries has probability at most $m^{-\omega(1)}$ of distinguishing between f and h .*

Consequently, the two functions f and h in Lemma 1 cannot be distinguished with high probability within a polynomial number of queries, i.e., within polynomial time. Hence, it suffices to construct two functions for which Lemma 1 holds.

Theorem 1. We will prove the bound in terms of the number $m = |\mathcal{E}|$ of edges in the graph. The graph we construct has $n = m - \ell + 2$ nodes, and therefore the proof also shows the lower bound in terms of nodes.

Construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with ℓ parallel disjoint paths from s to t , where each path has k edges. The random set $R \subset \mathcal{E}$ is always a cut consisting of $|R| = \ell$ edges, and contains one edge from each path uniformly at random. We define $\beta = 8\ell/k < \ell$ (for $k > 8$), and, for any $Q \subseteq \mathcal{E}$,

$$h(Q) = \min\{|Q|, \ell\} \tag{12}$$

$$f(Q) = \min\{|Q \setminus R| + \min\{|Q \cap R|, \beta\}, \ell\}. \tag{13}$$

The functions differ only for the relatively few sets Q with $|Q \cap R| > \beta$ and $|Q \setminus R| < \ell - \beta$, with $\min_{A \in \mathcal{C}} h(A) = h(C) = \ell$, $\min_{A \in \mathcal{C}} f(A) = f(R) = \beta$, where \mathcal{C} is the set of cuts, and C is any cut. We must have $k\ell = m$, so define ϵ such that $\epsilon^2 = 8/7 \log m$, and set $k = 8\sqrt{m}/\epsilon$ and $\ell = \epsilon\sqrt{m}/8$.

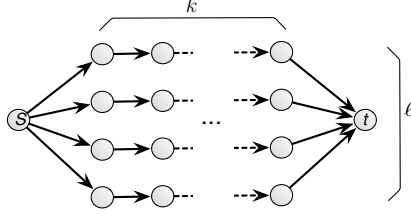


Figure 2: Graph for the proof of Theorem 1.

We compute the probability that f and h differ for a given query set Q . Probabilities are over the random unknown R . Since $f \leq h$, the probability of a difference is $P(f(Q) < h(Q))$. If $|Q| \leq \ell$, then $f(Q) < h(Q)$ only if $\beta < |Q \cap R|$, and the probability $P(f(Q) < h(Q)) = P(|Q \cap R| > \beta)$ increases as Q grows. If, on the other hand, $|Q| \geq \ell$, then since $h(Q) = \ell$ the probability

$$P(f(Q) < h(Q)) = P(|Q \setminus R| + \min\{|Q \cap R|, \beta\} < \ell) = P(|Q \cap R| > \beta)$$

decreases as Q grows. Hence, the probability of a difference is largest when $|Q| = \ell$.

So let $|Q| = \ell$. If Q spreads over $b \leq k$ edges of a path P , then the probability that Q includes the edge in $P \cap R$ is b/k . The expected overlap between Q and R is the sum of hits on all paths: $\mathbb{E}[|Q \cap R|] = |Q|/k = \ell/k$. Since the edges in R are independent across different paths, we may bound the probability of a large intersection by a Chernoff bound (with $\delta = 7$ in [51]):

$$P(f(Q) \neq h(Q)) \leq P(|Q \cap R| \geq 8\ell/k) \quad (14)$$

$$\leq 2^{-7\ell/k} = 2^{-7\epsilon^2/8} = 2^{-\omega(\log m)} = m^{-\omega(1)}. \quad (15)$$

With this result, Lemma 1 applies. No polynomial-time algorithm can guarantee to be able to distinguish f and h with high probability. A polynomial algorithm with approximation factor better than the ratio of optima $h(R)/f(R)$ would discriminate the two functions and thus lead to a contradiction. As a result, the lower bound is determined by the ratio of optima of h and f . The optimum of f is $f(R) = \beta$, and h has uniform cost ℓ for all minimal cuts. Hence, the ratio is $h(R)/f(R) = \ell/\beta = \sqrt{m}/\epsilon = o(\sqrt{m}/\log m)$. \square \square

Building on the construction in the above proof with $\ell = n^{1/3}$ and a different cut cost function, Balcan and Harvey [4] proved that if the data structure used by an algorithm (even with an arbitrary number of queries) has polynomial size, then this data structure cannot represent the minimizers of their cooperative cut problem to an approximation factor of $o(n^{1/3}/\log n)$.

In addition, we mention that a reduction from Graph Bisection serves to prove that MINCOOPCUT is NP-hard. We defer the proof to the appendix, but point out that in the reduction, the cost function is fully accessible and given as a polynomial-time computable formula.

Theorem 2. *Minimum Cooperative (s, t) -Cut is NP-hard.*

4 Relaxation and the Flow Dual

As a first step towards approximation algorithms, we formulate a relaxation of MINCOOPCUT and then address the associated flow-cut gap. The minimum cooperative cut problem can be relaxed to a continuous convex optimization problem using the convex Lovász extension \tilde{f} of f :

$$\begin{aligned} \min_{y \in \mathbb{R}^{|\mathcal{E}|}, x \in \mathbb{R}^{|\mathcal{V}|}} \quad & \tilde{f}(y) \\ \text{s.t.} \quad & -x(u) + x(v) + y(e) \geq 0 \quad \text{for all } e = (u, v) \in \mathcal{E} \\ & x(s) - x(t) \geq 1 \\ & y \geq 0 \end{aligned} \tag{16}$$

The dual of this problem can be derived by writing the Lovász extension as a maximum $\tilde{f}(y) = \max_{z \in \mathcal{P}(f)} z^\top y$ of linear functions. The maximum is taken over the submodular polyhedron

$$\mathcal{P}(f) = \{y \mid \sum_{e \in A} y(e) \leq f(A) \ \forall A \subseteq \mathcal{E}\}. \tag{17}$$

The resulting dual problem is a flow problem with non-local capacity constraints:

$$\max_{\nu \in \mathbb{R}, \varphi \in \mathbb{R}^{|\mathcal{E}|}} \quad \nu \tag{18}$$

$$\text{s.t.} \quad \varphi(A) \triangleq \sum_{e \in A} \varphi(e) \leq f(A) \quad \text{for all } A \subseteq \mathcal{E} \tag{19}$$

$$\begin{aligned} \sum_{e \in \delta^+ u} \varphi(e) - \sum_{e' \in \delta^- u} \varphi(e') &= d(u)\nu \quad \text{for all } u \in \mathcal{V} \\ \varphi &\geq 0, \end{aligned}$$

where $d(u) = 1$ if $u = s$, $d(u) = -1$ if $u = t$, and $d(u) = 0$ otherwise. Constraint (19) demands that φ must, in addition to satisfying the common flow conservation, reside within the submodular polyhedron $\mathcal{P}(f)$. This more restrictive constraint replaces the edge-wise capacity constraints that occur when f is a sum of weights. We name this problem MAXIMUM COOPERATIVE FLOW.

Alternatively to (16), the constraints can be stated in terms of paths: a set of edges is a cut if it intersects all (s, t) -paths in the graph.

$$\begin{aligned} \min \quad & \tilde{f}(y) \\ \text{s.t.} \quad & \sum_{e \in P} y(e) \geq 1 \quad \text{for all } (s, t)\text{-paths } P \subseteq \mathcal{E} \\ & y \in [0, 1]^\mathcal{E}. \end{aligned} \tag{20}$$

We will use this form in Section 5.2.1, and use the relaxation (16) in Section 5.2.2.

4.1 Flow-cut Gap

The relaxation (16) of the discrete problem (1) is not tight. This becomes evident when analyzing the ratio $f(C^*)/\hat{f}(y^*)$ between the optimal value of the discrete problem and the relaxation (16) (the *integrality gap*). This ratio is, by strong duality between Problems (16) and (18), the flow-cut gap $f(C^*)/\nu^*$ of the optimal cut and maximal flow values.

Lemma 2. *Let \mathcal{P} be the set of all (s, t) -paths in the graph. The flow-cut gap $f(C^*)/\nu^*$ can be upper and lower bounded as*

$$\frac{f(C^*)}{\sum_{P \in \mathcal{P}} \min_{P' \subseteq P} \frac{f(P')}{|P'|}} \leq \frac{f(C^*)}{\nu^*} \leq \frac{f(C^*)}{\max_{P \in \mathcal{P}} \min_{P' \subseteq P} \frac{f(P')}{|P'|}}$$

Proof. The Lemma straightforwardly follows from bounding the optimal flow ν^* . The flow through a single path $P \in \mathcal{P}$, if all other edges $e \notin P$ are empty, is restricted by the minimum average capacity for any subset of edges within the path, i.e., $\min_{P' \subseteq P} \frac{f(P')}{|P'|}$. Moreover, we obtain a family of feasible solutions as those that send nonzero flow only along one path and remain within that path's capacity. Hence, the maximum flow must be at least as big as the flow for any of those single-path solutions. This observation yields the upper bound on the ratio.

A similar argumentation shows the lower bound: The total joint capacity constraint is upper bounded by $\hat{f}(A) = \sum_{P \in \mathcal{P}} f(A \cap P) \geq f(A)$. Hence, $\sum_{P \in \mathcal{P}} \min_{P' \subseteq P} \frac{f(P')}{|P'|}$ is the value of the maximum flow with capacity \hat{f} if each edge is only contained in one path, and is an upper bound on the flow otherwise. \square \square

The gap in Lemma 2 can be large:

Corollary 1. *The flow-cut gap for MINCOOPCUT can be as large as $n - 1$.*

Proof. Corollary 1 can be shown via an example where the upper and lower bound of Lemma 2 coincide. The worst-case example for the flow-cut gap is a simple graph that consists of one single path from s to t with $n - 1$ edges. For this graph one of the capacity constraints is that

$$\varphi(\mathcal{E}) = \sum_{e \in \mathcal{E}} \varphi(e) \leq f(\mathcal{E}). \quad (21)$$

Constraint (21) is the only relevant capacity constraint if the capacity (and cut cost) function is $f(A) = \max_{e \in A} w(e)$ with weights $w(e) = \gamma$ for all $e \in \mathcal{E}$ and some constant $\gamma > 0$ and, consequently, $f(\mathcal{E}) = \gamma$. By Constraint (21), the maximum flow is $\nu^* = \frac{\gamma}{n-1}$. The optimum cooperative cut C^* , by contrast, consists of any single edge and has cost $f(C^*) = \gamma$. \square \square

Single path graphs as used in the previous proof can provide worst-case examples for rounding methods too: if f is such that $f(e) \geq f(\mathcal{E})/|\mathcal{E}|$ for all edges e in the path, then the solution to the relaxed cut problem is maximally uninformative: all entries of the vector y are $y(e) = \frac{f(\mathcal{E})}{n-1}$.

approximating f		relaxation	
generic (§5.1.1)	$O(\sqrt{m} \log m)$	randomized (§5.2.1)	$ P_{\max} $
semigradient (§5.1.2)	$\frac{ C^* }{(C^* -1)(1-\kappa_f)+1}$	rounding I (§5.2.2)	$ P_{\max} $
polymatroidal flow (§5.1.3)	$\min\{\Delta_s, \Delta_t\}$	rounding II (§5.2.2)	$ \mathcal{V} - 1$

Table 2: Overview of the algorithms and their approximation factors.

5 Approximation Algorithms

We next address approximation algorithms, whereby we consider two complementary approaches. The first approach is to substitute the submodular cost function f by a simpler function \hat{f} . Appropriate candidate functions \hat{f} that admit an exact cut optimization are the approximation by Goemans et al. [23] (Section 5.1.1), semi-gradient based approximations (Section 5.1.2), or approximations by making f separable across local neighborhoods (Section 5.1.3).

The second approach is to solve the relaxations from Section 4 and round the resulting optimal fractional solution (Section 5.2.2). Conceptually very close to the relaxations is an algorithm that solves the mathematical program (20) via a randomized greedy algorithm (Section 5.2.1).

The relaxations approaches are affected by the flow-cut gap, or, equivalently, the length of the longest path in the graph. The approximations that use a surrogate cost function are complementary and not affected by the “length”, but by a notion of the “width” of the graph.

5.1 Approximating the cost function

We begin with algorithms that use a suitable approximation \hat{f} to f , for which the problem

$$\text{minimize } \hat{f}(C) \quad \text{s.t. } C \subseteq \mathcal{E} \text{ is a cut} \quad (22)$$

is solvable exactly in polynomial time. The following lemma will be the basis for approximation bounds.

Lemma 3. *Let $\hat{S} = \operatorname{argmin}_{S \in \mathcal{S}} \hat{f}(S)$. If for all $S \subseteq \mathcal{E}$, it holds that $f(S) \leq \hat{f}(S)$, and if for the optimal solution S^* to Problem (1), it holds that $\hat{f}(S^*) \leq \alpha f(S^*)$, then \hat{S} is an α -approximate solution to Problem (1):*

$$f(\hat{S}) \leq \alpha f(S^*).$$

Proof. Since $\hat{f}(\hat{S}) \leq \hat{f}(S^*)$, it follows that $f(\hat{S}) \leq \hat{f}(\hat{S}) \leq \hat{f}(S^*) \leq \alpha f(S^*)$. \square

5.1.1 A generic approximation

Goemans et al. [23] define a generic approximation of a submodular function⁴ that has the functional form $\hat{f}_{ea}(A) = \sqrt{\sum_{e \in A} w_f(e)}$. The weights $w_f(e)$ de-

⁴We will also call it the *ellipsoidal approximation* since it is based on approximating a symmetrized version of the submodular polyhedron by an ellipsoid.

pend on f . When using \hat{f}_{ea} , we compute a minimum cut for the cost \hat{f}_{ea}^2 , which is a modular sum of weights and hence results in a standard MINIMUM (s, t) -CUT problem. In practice, the bottleneck lies in computing the weights w_f . Goemans et al. [23] show how to compute weights such that $f(A) \leq \hat{f}(A) \leq \alpha f(A)$ with $\alpha = O(\sqrt{m})$ for a matroid rank function, and $\alpha = O(\sqrt{m} \log m)$ otherwise. We add that for an integer polymatroid rank function bounded by $M = \max_{e \in \mathcal{E}} f(e)$, the logarithmic factor can be replaced by a constant to yield $\alpha = O(\sqrt{mM})$ (if one approximates the matroid expansion⁵ of the polymatroid instead of f directly). Together with Lemma 3, this yields the following approximation bounds.

Lemma 4. *Let $\hat{C} = \operatorname{argmin}_{C \in \mathcal{C}} \hat{f}_{ea}(C)$ be the minimum cut for cost \hat{f}_{ea} , and $C^* = \operatorname{argmin}_{C \in \mathcal{C}} f(C)$. Then $f(\hat{C}) = O(\sqrt{m} \log m) f(C^*)$. If f is integer-valued and we approximate its matroid expansion, then $f(\hat{C}) = O(\sqrt{mM}) f(C^*)$, where $M \leq \max_e f(e)$.*

The lower bound in Theorem 1 suggests that for sparse graphs, the bound in Lemma 4 is tight up to logarithmic factors.

5.1.2 Approximations via semigradients

For any monotone submodular function f and any set A , there is a simple way to compute a modular upper bound \hat{f}_s to f that agrees with f at A . In other words, \hat{f}_s is a discrete *supergradient* of f at A . We define \hat{f}_s as [32, 29]

$$\hat{f}_s(B; A) = f(A) + \sum_{e \in B \setminus A} f(e \mid A) - \sum_{e \in A \setminus B} f(e \mid \mathcal{E} \setminus e). \quad (23)$$

Lemma 5. *Let $\hat{C} \in \operatorname{argmin}_{C \in \mathcal{C}} \hat{f}_s(C; \emptyset)$. Then*

$$f(\hat{C}) \leq \frac{|C^*|}{(|C^*| - 1)(1 - \kappa_f) + 1} f(C^*),$$

where $\kappa_f = \max_e 1 - \frac{f(e \mid \mathcal{E} \setminus e)}{f(e)}$ is the curvature of f .

Lemma 5 was shown in [29]. As m (and correspondingly $|C^*|$) gets large, the bound eventually no longer depends on m and instead only on the curvature of f . In practice, results are best when the supergradient is used in an iterative algorithm: starting with $C_0 = \emptyset$, one computes $C_t \in \operatorname{argmin}_{C \in \mathcal{C}} \hat{f}_s(C; C_{t-1})$ until the solution no longer changes between iterations. The minimum cut for the cost function $\hat{f}_s(C; A)$ can be computed as a minimum cut with edge weights

$$w(e) = \begin{cases} f(e \mid \mathcal{E}) & \text{if } e \in A \\ f(e \mid A) & \text{if } e \notin A. \end{cases} \quad (24)$$

⁵The expansion is described in Section 10.3 in [52]. In short, we replace each element e by a set \hat{e} of $f(e)$ parallel elements. Thereby we extend f to a submodular function \hat{f} on subsets of $\bigcup_i \hat{e}_i$. The desired rank function is now the convolution $r(\cdot) = \hat{f}(\cdot) * |\cdot|$ and it satisfies $f(S) = r(\bigcup_{e \in S} \hat{e})$.

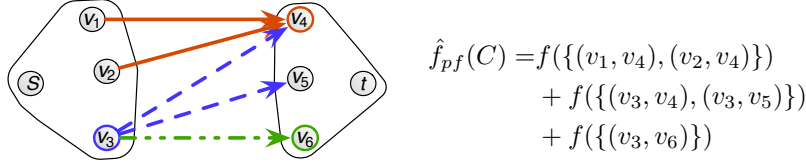


Figure 3: Approximation of a cut cost. Red edges are in $C_{v_4}^\Pi$ (head), blue dashed edges in $C_{v_3}^\Pi$ (tail), and the green dash-dotted edge in $C_{v_6}^\Pi$ (head).

Consequently, the semigradient approximation yields a very easy and practical algorithm that iteratively uses standard minimum cut as a subroutine. This algorithm was used e.g. in [32], and the visual results in [42] show that often, it can yield practically good solutions.

5.1.3 Approximations by Introducing Separation

The approximations in Section 5.1.1 and 5.1.2 are indifferent to the structure of the graph. The following approximation is not. One may say that Problem (2) is hard because f introduces non-local dependencies between edges anywhere in the graph. Indeed, the problem is easier if dependencies are restricted to local neighborhoods.

Hence, we define an approximation \hat{f}_{pf} that is globally separable but locally exact. To measure the cost of an edge set $C \subseteq \mathcal{E}$, we partition C into groups $\Pi(C) = \{C_v^\Pi\}_{v \in V}$, where the edges in set C_v^Π must be incident to node v (C_v^Π may be empty). That is, we assign each edge either to its head or to its tail node, as illustrated in Figure 3. Let \mathcal{P}_C be the family of all such partitions (which vary over the head or tail assignment of each edge). We define an approximation

$$\hat{f}_{pf}(C) = \min_{\Pi(C) \in \mathcal{P}_C} \sum_{v \in V} f(C_v^\Pi) \quad (25)$$

that (once the partition is fixed) decomposes across different node incidence edge sets, but is accurate within a group C_v^Π . Thanks to the subadditivity of f , the function \hat{f}_{pf} is an upper bound on f . It is a convolution of submodular functions and always is the tightest approximation that is a direct sum over any partition in \mathcal{P}_C . Even though the approximation (25) looks difficult to compute and is in general not even a submodular function (shown by the example in Appendix C), it is possible to solve a minimum cut with cost \hat{f}_{pf} exactly. To do so, we exploit its duality to a generalized maximum flow problem, namely polymatroidal network flows.

Polymatroidal network flows. Polymatroidal network flows [48, 26] generalize the capacity constraint of traditional flow problems. They retain the constraint of flow conservation (a function $\varphi : \mathcal{E} \rightarrow \mathbb{R}_+$ is a flow if the inflow at each node $v \in \mathcal{V} \setminus \{s, t\}$ equals the outflow). The edge-wise capacity constraint $\varphi(e) \leq \text{cap}(e)$ for all $e \in \mathcal{E}$, given a capacity function $\text{cap} : \mathcal{E} \rightarrow \mathbb{R}_+$ is replaced

by local submodular capacities over *sets* of edges incident at each node v : cap_v^{in} for incoming edges, and $\text{cap}_v^{\text{out}}$ for outgoing edges. The capacity constraints at each $v \in \mathcal{V}$ are

$$\begin{aligned} \varphi(A) &\leq \text{cap}_v^{\text{in}}(A) && \text{for all } A \subseteq \delta^-(v) \text{ (incoming edges), and} \\ \varphi(A) &\leq \text{cap}_v^{\text{out}}(A) && \text{for all } A \subseteq \delta^+(v) \text{ (outgoing edges).} \end{aligned}$$

Each edge (u, v) belongs to two incidence sets, δ^+u and δ^-v . A maximum flow with such constraints can be found in time $O(m^4\tau)$ by the layered augmenting path algorithm by Tardos et al. [63], where τ is the time to minimize a submodular function on any set δ^+v, δ^-v . The incidence sets are in general much smaller than \mathcal{E} .

A special polymatroidal maximum flow turns out to be dual to the cut problem we are interested in. To see this, we will use the restriction $f|_A$ of the function f to a subset A . For ease of reading we drop the explicit restriction notation later. We assume throughout that the desired cut is minimal⁶, since additional edges can only increase its cost.

Lemma 6. *Minimum (s, t) -cut with cost function \hat{f}_{pf} is dual to a polymatroidal network flow with capacities $\text{cap}_v^{\text{in}} = f|_{\delta^-v}$ and $\text{cap}_v^{\text{out}} = f|_{\delta^+v}$ at each node $v \in \mathcal{V}$.*

The proof is provided in Appendix D. It uses, with some additional considerations, the dual problem to a polymatroidal maxflow, which can be stated as follows. Let $\text{cap}^{\text{in}} : 2^{\mathcal{E}} \rightarrow \mathbb{R}_+$ be the joint incoming capacity function, i.e., $\text{cap}^{\text{in}}(C) = \sum_{v \in V} \text{cap}_v^{\text{in}}(C \cap \delta^-v)$, and let equivalently cap^{out} be the corresponding joint outgoing capacity. The dual of the polymatroidal maximum flow is a minimum cut problem whose cost is a convolution of edge capacities [49]:

$$\text{cap}(C) = (\text{cap}^{\text{in}} * \text{cap}^{\text{out}})(C) \triangleq \min_{A \subseteq C} [\text{cap}^{\text{in}}(A) + \text{cap}^{\text{out}}(C \setminus A)]. \quad (26)$$

This convolution is in general not a submodular function. Lemma 6 implies that we can solve the approximate MINCOOPCUT via its dual flow problem. The primal cut solution will be given by a set of full edges, i.e., edges whose joint flow equals their joint capacity.

We can now state the resulting approximation bound for MINCOOPCUT. Let C^* be the optimal cut for cost f . We define Δ_s to be the tail nodes of the edges in C^* : $\Delta_s = \{v \mid \exists (v, u) \in C^*\}$, and similarly, $\Delta_t = \{v \mid \exists (u, v) \in C^*\}$. The sets Δ_s, Δ_t provide a measure of the “width” of the graph.

Theorem 3. *Let \hat{C} be the minimum cut for cost \hat{f}_{pf} , and C^* the optimal cut for cost f . Then*

$$f(\hat{C}) \leq \min\{|\Delta_s|, |\Delta_t|\} f(C^*) \leq \frac{|\mathcal{V}|}{2} f(C^*).$$

⁶A cut $C \subseteq \mathcal{E}$ is *minimal* if no proper subset $B \subset C$ is a cut.

Proof. To apply Lemma 3, we need to show that $f(C) \leq \hat{f}_{pf}(C)$ for all $C \subseteq \mathcal{E}$, and find an α such that $\hat{f}_{pf}(C^*) \leq \alpha f(C^*)$. The first condition follows from the subadditivity of f .

To bound α , we use Lemma 6 and Equation 26:

$$\hat{f}_{pf}(C^*) = (\text{cap}^{\text{in}} * \text{cap}^{\text{out}})(C^*) \quad (27)$$

$$\leq \min\{\text{cap}^{\text{in}}(C^*), \text{cap}^{\text{out}}(C^*)\} \quad (28)$$

$$\leq \min\left\{\sum_{v \in \Delta_s} f(C^* \cap \delta^+ v), \sum_{v \in \Delta_t} f(C^* \cap \delta^- v)\right\} \quad (29)$$

$$\leq \min\left\{|\Delta_s| \max_{v \in \Delta_s} f(C^* \cap \delta^+ v), |\Delta_t| \max_{v \in \Delta_t} f(C^* \cap \delta^- v)\right\} \quad (30)$$

$$\leq \min\{|\Delta_s|, |\Delta_t|\} f(C^*). \quad (31)$$

Thus, Lemma 3 implies an approximation bound $\alpha \leq \min\{|\Delta_s|, |\Delta_t|\} \leq |\mathcal{V}|/2$. \square

Iyer et al. [30] show that the bound in Theorem 3 can be tightened to $\frac{|\mathcal{V}|}{2 + (|\mathcal{V}| - 2)(1 - \kappa_f)}$ by taking into account the curvature κ_f of f .

5.2 Relaxations

An alternative approach to approximating the edge weight function f is to relax the cut constraints via the formulations (20) and (16). We analyze two algorithms: the first, a randomized algorithm, maintains a discrete solution, while the second is a simple rounding method. Both cases remove the constraint that the cut must be minimal: any set B is feasible that has a *subset* $C \subseteq B$ that is a cut. Relaxing the minimality constraint makes the feasible set *up-monotone* (equivalently up-closed). This is not major problem, however, since any superset of a cut can easily be pruned to a minimal cut.

5.2.1 Randomized greedy covering

The constraints in the path-based relaxation (20) suggest that a minimum (s, t) -cut problem is also a min-cost cover problem: a cut must intersect or “cover” each (s, t) -path in the graph. The covering formulation of the constraints in (20) clearly show the relaxation of the minimality constraint. Algorithm 1 solves a discrete variant of the formulation (20) and maintains a discrete $y \in \{0, 1\}$, i.e., y is eventually the *indicator vector* of a cut.

Since a graph can have exponentially many (s, t) -paths, there can be exponentially many constraints. But all that is needed in the algorithm is to find a violated constraint, and this is possible by computing the shortest path P_{\min} , using y as the (additive) edge lengths. If P_{\min} is longer than one, then y is feasible. Otherwise, P_{\min} defines a violated constraint.

Owing to the form of the constraints, we can adapt a randomized greedy cover algorithm [46] to Problem (20) and obtain Algorithm 1. In each step, we compute the shortest path with weights y to find a possibly uncovered path. Ties

Algorithm 1 Greedy randomized path cover

Input: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, terminal nodes $s, t \in \mathcal{V}$, cost function $f : 2^{\mathcal{E}} \rightarrow \mathbb{R}_+$

```

 $C = \emptyset, y = 0$ 
while  $\sum_{e \in P_{\min}} y(e) < 1$  for the shortest path  $P_{\min}$  do
  choose  $\beta$  within the interval  $\beta \in (0, \min_{e \in P_{\min}} f(e|C)]$ 
  for  $e$  in  $P_{\min}$  do
    with probability  $\beta/f(e|C)$ , set  $C = C \cup \{e\}$ ,  $y(e) = 1$ .
  end for
end while
prune  $C$  to  $C'$  and return  $C'$ 

```

are resolved arbitrarily. To cover the path, we randomly pick edges from P_{\min} . The probability of picking edge e is inversely proportional to the marginal cost $f(e|C)$ of adding e to the current selection of edges⁷. We must also specify an appropriate β . With the maximal allowed $\beta = \min_{e \in P_{\min}} f(e|C)$, the cheapest edges are selected deterministically, and others randomly. In that case, C grows by at least one edge in each iteration, and the algorithm terminates after at most m iterations.

If the algorithm returns a set C that is feasible but not a *minimal* cut, it is easy to prune it to a minimal cut $C' \subseteq C$ without any additional approximation error, since monotonicity of f implies that $f(C') \leq f(C)$. Such pruning can for example be done via breadth-first search. Let \mathcal{V}_s be the set of nodes reachable from s after the edges in C have been removed. Then we set $C' = \delta(\mathcal{V}_s)$. The set C' must be a subset of C : if there was an edge $(u, v) \in C' \setminus C$, then v would also be in \mathcal{V}_s , and then (u, v) cannot be in C' , a contradiction.

The approximation bound for Algorithm 1 is the length of the longest path, like that of the rounding methods in Section 5.2.2. This is not a coincidence, since both algorithms essentially use the same relaxation.

Lemma 7. *In expectation (over the probability of sampling edges), Algorithm 1 returns a solution \hat{C}' with $\mathbb{E}[f(\hat{C}')] \leq |P_{\max}|f(C^*)$, where P_{\max} is the longest simple (s, t) -path in \mathcal{G} .*

Proof. Let \hat{C} be the cut before pruning. Since f is nondecreasing, it holds that $f(\hat{C}') \leq f(\hat{C})$. By Theorem 7 in [46], a greedy randomized procedure like Algorithm 1 yields in expectation an α -approximation for a cover, where α is the maximum number of variables in any constraint. Here, α is the maximum number of edges in any simple path, i.e., the length of the longest path. This implies that $\mathbb{E}[f(\hat{C}')] \leq \mathbb{E}[f(\hat{C})] \leq |P_{\max}|f(C^*)$. \square \square

Indeed, randomization is important. Consider a deterministic algorithm that always picks the edge with minimum marginal cost in the next path to cover.

⁷If $\min_{e \in P_{\min}} f(e|C) = 0$, then we greedily pick all such edges with zero marginal cost, because they do not increase the cost. Otherwise we sample as indicated in the algorithm.

The solution \hat{C}_d returned by this algorithm can be much worse. As an example, consider a graph consisting of a clique \mathcal{V} of n nodes, with nodes s and t . Let $S \subseteq \mathcal{V}$ be a set of size $n/2$. Node s is connected to all nodes in S , and node t is connected to the clique only by a distinct node $v' \in \mathcal{V} \setminus S$ via edge (v', t) . Let the cost function be a sum of edge weights, $f(C) = \sum_{e \in C} w(e)$. Edge (v', t) has weight $\gamma > 0$, all edges in $\delta^+(S)$ have weight $\gamma(1 - \epsilon)$ for a small $\epsilon > 0$, and all remaining edges have weight $\gamma(1 - \epsilon/2)$. The deterministic algorithm will return $\hat{C}_d = \delta^+(S)$ as the solution, with cost $\frac{n^2\gamma}{4}(1 - \epsilon)$, which is by a factor of $|\hat{C}_d|(1 - \epsilon) = \frac{n^2}{4}(1 - \epsilon)$ worse than the optimal cut, $f(\{(v', t)\}) = \gamma$. Hence, for the deterministic variant of Algorithm 1, we can only show the following approximation bound:

Lemma 8. *For the solution \hat{C}_d returned by the greedy deterministic heuristic, it holds that $f(\hat{C}_d) \leq |\hat{C}_d|f(C^*)$. This approximation factor cannot be improved in general.*

Proof. To each edge $e \in \hat{C}_d$ assign the path $P(e)$ which it was chosen to cover. By the nature of the algorithm, it must hold that $f(e) \leq f(C^* \cap P(e))$, because otherwise an edge in $C^* \cap P(e)$ would have been chosen. Since C^* is a cut, the set $C^* \cap P(e)$ must be non-empty. These observations imply that

$$f(\hat{C}_d) \leq \sum_{e \in \hat{C}_d} f(e) \leq \sum_{e \in \hat{C}_d} f(C^* \cap P(e)) \leq |\hat{C}_d| \max_{e \in \hat{C}_d} f(C^* \cap P(e)) \leq |\hat{C}_d|f(C^*).$$

Tightness follows from the worst-case example described above. \square \square

5.2.2 Rounding

Our last approach is to solve the convex program (16) and round the continuous to a discrete solution. We describe two types of rounding, each of which achieves a worst-case approximation factor of $n - 1$. This factor equals the general flow-cut gap in Lemma 1. Let x^*, y^* be the optimal solution to the relaxation (16) (equivalently, to (20)). We assume w.l.o.g. that $x^* \in [0, 1]^n$, $y^* \in [0, 1]^m$.

Rounding by thresholding edge lengths. The first technique uses the edge weights y^* . We pick a threshold θ and include all edges e whose entry $y^*(e)$ is larger than θ . Algorithm 2 shows how to select θ , namely the largest edge length that when treated as a threshold yields a cut.

Lemma 9. *Let \hat{C} be the rounded solution returned by Algorithm 2, θ the threshold at the last iteration i , and C^* the optimal cut. Then*

$$f(\hat{C}) \leq \frac{1}{\theta}f(C^*) \leq |P_{\max}|f(C^*) \leq (n - 1)f(C^*),$$

where P_{\max} is the longest simple path in the graph.

Algorithm 2 Rounding procedure given y^*

order \mathcal{E} such that $y^*(e_1) \geq y^*(e_2) \geq \dots \geq y^*(e_m)$
for $i = 1, \dots, m$ **do**
 let $C_i = \{e_j \mid y^*(e_j) \geq y^*(e_i)\}$
 if C_i is a cut **then**
 prune C_i to \hat{C} and return \hat{C}
 end if
end for

Proof. The proof is analogous to that for covering problems [28]. In the worst case, y^* is uniformly distributed along the longest path, i.e., $y^*(e) = |P_{\max}|^{-1}$ for all $e \in P_{\max}$ as y^* must sum to at least one along each path. Then θ must be at least $|P_{\max}|^{-1}$ to include at least one of the edges in P_{\max} . Since \tilde{f} is nondecreasing like f and also positively homogeneous, it holds that

$$f(\hat{C}) \leq f(C_i) = \tilde{f}(\chi_{C_i}) \leq \tilde{f}(\theta^{-1}y^*) = \theta^{-1}\tilde{f}(y^*) \leq \theta^{-1}\tilde{f}(\chi_{C^*}) = \theta^{-1}f(C^*).$$

The first inequality follows from monotonicity of f and the fact that $\hat{C} \subseteq C_i$. Similarly, the relation between $\tilde{f}(\chi_{C_i})$ and $\tilde{f}(\theta^{-1}y^*)$ holds because \tilde{f} is nondecreasing: by construction, $y^*(e) \geq \theta\chi_{C_i}(e)$ for all $e \in \mathcal{E}$, and hence $\chi_{C_i}(e) \leq \theta^{-1}y^*(e)$. Finally, we use the optimality of y^* to relate the cost to $f(C^*)$; the vector χ_{C^*} is also feasible, but y^* optimal. The lemma follows since $\theta^{-1} \leq |P_{\max}|$. \square \square

Rounding by node distances. Alternatively, we can use x^* to obtain a discrete solution. We pick a threshold θ uniformly at random from $[0, 1]$ (or find the best one), and choose all nodes u with $x^*(u) \geq \theta$ (call this \mathcal{V}_θ) and use the cut $C_\theta = \delta(\mathcal{V}_\theta)$. As the node labels x^* can also be considered as distances from s , we will refer to these rounding methods as *distance-based rounding*.

Lemma 10. *The worst-case approximation factor for a solution C_θ obtained with distance-based rounding is $\mathbb{E}_\theta[f(C_\theta)] \leq (n-1)\tilde{f}(y^*) \leq (n-1)f(C^*)$.*

Proof. To upper bound the quantity $\mathbb{E}_\theta[f(C_\theta)]$, we partition the set of edges into $(n-1)$ sets $\delta^+(v)$, that is, each set corresponds to the outgoing edges of a node $v \in \mathcal{V}$. We sort the edges in each $\delta^+(v)$ in nondecreasing order by their values $y^*(e)$. Consider one specific incidence set $\delta^+(u)$ with edges $e_{u,1}, \dots, e_{u,h}$ and $y^*(e_{u,1}) \leq y^*(e_{u,2}) \leq \dots \leq y^*(e_{u,h})$. Edge $e_{u,i}$ is in the cut if $\theta \in [x^*(u), x^*(u) + y^*(e_{u,i}))$. Therefore, it holds for each node u that

$$\mathbb{E}_\theta[f(C_\theta \cap \delta^+(u))] = \int_0^1 f(C_\theta \cap \delta^+(u)) d\theta \quad (32)$$

$$= \sum_{j=1}^h (y^*(e_{u,j}) - y^*(e_{u,j-1})) f(\{e_{u,j}, \dots, e_{u,h}\}) \quad (33)$$

$$= \tilde{f}(y^*(\delta^+(u))), \quad (34)$$

where we define $y^*(e_{u,0}) = 0$ for convenience, and assume that $f(\emptyset) = 0$. This implies that

$$\mathbb{E}_\theta[f(C_\theta)] \leq \mathbb{E}_\theta\left[\sum_{v \in \mathcal{V}} f(C_\theta \cap \delta^+(v))\right] \quad (35)$$

$$= \sum_{v \in \mathcal{V}} \tilde{f}(y^*(\delta^+(v))) \leq (n-1)\tilde{f}(y^*) \leq (n-1)f(C^*). \quad (36)$$

□

□

A more precise approximation factor is $\frac{\sum_v \tilde{f}(y^*(\delta^+(v)))}{f(y^*)}$.

6 Special cases

The complexity of MINCOOPCUT is not always as bad as the worst-case bound in Theorem 1. We next discuss properties of the submodular cost function and the graph structure that lead to better approximation factors. Our discussion is not specific to cooperative cuts; it is rather a survey of properties that make a number of submodular optimization problems easier.

6.1 Separability

An important factor for tractability and approximations is the separability of the cost function, that is, whether there are *separators* of f whose structure aligns with the graph.

Definition 1 (Separator of f). *A set $S \subseteq \mathcal{E}$ is called a separator of $f : 2^\mathcal{E} \rightarrow \mathbb{R}$ if for all $B \subseteq \mathcal{E}$, it holds that $f(B) = f(B \cap S) + f(B \setminus S)$. The set of separators of f is closed under union and intersection.*

The structure of the separators strongly affects the complexity⁸ of MINCOOPCUT. First and obviously, the extreme case that f is a modular function (and each $e \in \mathcal{E}$ is a separator) can be solved exactly. Second, if the separators of f form a partition $\mathcal{E} = \bigcup_v E_v^+ \cup \bigcup_v E_v^-$ that aligns with node neighborhoods such that $E_v^+ \subseteq \delta^+(v)$, and $E_v^- \subseteq \delta^-(v)$, then both \hat{f}_{pf} and distance-based rounding solve the problem exactly. In that case, the flow-cut gap is zero, as becomes obvious from the proof of Lemma 10, since $(\sum_v \tilde{f}(y_{E_v}^*)) / \tilde{f}(y^*) = 1$. These separators respect the graph structure and rule out any non-local edge interactions.

6.2 Symmetry and “unstructured” functions

A submodular function with disjoint separators $\{B_i\}_i$ ($\mathcal{V} = \bigcup_i B_i$) can be written as a sum $f(A) = \sum_i f(A \cap B_i)$ of submodular functions, each of which has

⁸ Assuming that f is computable in polynomial time (or given by an oracle) and that we have access to the set of separators.

support only on one separator. If the restricted functions are “easy” because the separators B_i are local and small as above, then MINCOOPCUT can be solved exactly, or with a better approximation factor.

Similarly, one may consider other sums of other functions that may not have separators beyond \mathcal{E} but are of a simpler form. An important class of such simple functions are those of the form $f_i(A) = g(\sum_{e \in A} w_i(e))$ for nonnegative weights $w_i(e)$ and a nondecreasing concave function g . We refer to the submodular functions $g(w(A))$ as *unstructured*, because they only consider a sum of weights, but otherwise do not make any distinction between edges (unlike, e.g., graphic matroid rank functions). One may classify such functions into a hierarchy, where $\mathcal{F}(k)$ contains all functions $f(A) = \sum_{j=1}^k g_j(w_j(A))$ with at most k such components. The functions $\mathcal{F}(k)$ are special cases of low-rank quasi-concave functions, where k is the rank of the function.

If $k = 1$, then it suffices to minimize $w_1(C)$ directly and the problem reduces to MINIMUM (s, t) -CUT. For $k > 1$, several combinatorial problems admit an FPTAS with running time exponential in k [24, 50]. This holds for cooperative cuts too [42]. A special case for $k = 2$ is the *mean-risk* objective $f(A) = w_1(A) + \sqrt{w_2(A)}$ [53]. Goel et al. [21] show that these functions may yield better bounds in combinatorial multi-agent problems than general polymatroid rank functions, if each agent has a cost function in $\mathcal{F}(1)$.

Even for general, unconstrained submodular minimization⁹, the class $\mathcal{F}(k)$ admits specialized improved optimization algorithms [40, 61, 43, 35]. The complexity of those faster specialized algorithms too depends on the rank k . An interesting question arising from the above observations is whether $\mathcal{F}(k)$ contains *all* submodular functions, if k is large enough? The answer is no: even if k is allowed to be exponential, this class is a strict sub-class of all submodular functions. If the addition of auxiliary variables is allowed, this class coincides with the class of graph-representable functions in the sense of [67]: any graph cut function $h : 2^V \rightarrow \mathbb{R}_+$ is in $\mathcal{F}(|\mathcal{E}|)$, and any function in $\mathcal{F}(k)$ can be represented as a graph cut function in an extended auxiliary graph [34]. However, not all submodular functions can be represented in this way [67].

The parameter k is a measure of complexity. If k is not fixed, then MINCOOPCUT is NP-hard; for example, the reduction in Section B uses such functions. Even more, unrestricted k may induce large lower bounds, as has been proved for *label cost* functions of the form $f(A) = \sum_{j=1}^k w_j \min\{1, |A \cap B_j|\}$ [66].

A subclass of unstructured submodular functions are *symmetric* submodular functions¹⁰ in the sense that they are indifferent to the identity of the elements: let σ be a permutation of the ground set, and $\sigma(A)$ the elements chosen from the permuted indices. A submodular function is symmetric if $f(A) = f(\sigma(A))$ for all permutations σ . Examples of such functions are functions that only depend on the cardinality of the argument. This symmetry affects the complexity of sub-

⁹For unconstrained submodular function minimization we drop the constraint that the functions g_j are nondecreasing.

¹⁰These are distinct from the other previously-used notion of symmetric submodular functions Queyranne [54] where, for all $A \subseteq \mathcal{E}$, $f(A) = f(\mathcal{E} \setminus A)$.

modular optimization problems, and its influence on submodular maximization problems has been studied by Vondrák [65].

6.3 Symmetry and graph structure

For symmetric submodular functions and clique graphs, the node function $h(X) = f(\delta^+(X))$ induced by cooperative cuts can be submodular for subsets $X \subseteq \mathcal{V}$. This special case depends on both the graph and the cost function. Examples of such cases are the P^n Potts functions and robust P^n potentials in [39, 41], which correspond to cost functions of the form $f(A) = g(|A|)$ [32]. Similarly, the total variation terms in [12] that are summarized in Section 2 are relaxations of cooperative cuts that generate submodular functions [31, Ch. 6]. These cost functions are both locally separable and symmetric on the local separators.

6.4 Curvature

The *curvature* $\kappa_f \in [0, 1]$ of a submodular function f is defined as

$$\kappa_f = \max_{e \in \mathcal{E}} 1 - \frac{f(e \mid \mathcal{E} \setminus e)}{f(e)}, \quad (37)$$

and characterizes the deviation from being a modular function. Curvature is known to affect the approximation bounds for submodular maximization [14, 64], and also for submodular minimization problems, approximating and learning submodular functions [30]. The lower the curvature, the better the approximation factors. For MINCOOPCUT and many other combinatorial minimization problems with submodular costs, the approximation factor is affected as follows. If α_n is the worst-case factor (e.g., for the semigradient approximation), then the tightened factor is $\frac{\alpha_n}{(\alpha_n - 1)(1 - \kappa_f) + 1}$. The lower bounds can be tightened accordingly.

6.5 Flow-cut gaps revisited

The above properties that facilitate MINCOOPCUT reduce the flow-cut gaps in some cases. The proof of Lemma 1 illustrates that the flow-cut gap is intricately linked to the edge cooperation (non-separability) along paths in the graph. Therefore, the separability described in Section 6.1 affects the flow-cut gap if it breaks up cooperation along paths: the gap depends only on the longest cooperating path within any separator of f , and this can be much smaller than n . If, however, an instance of MINCOOPCUT is better solvable because the cost function is a member of $\mathcal{F}(\ell)$ for small constant ℓ , then the gap may still be as large as in Lemma 1. In fact, the example in Lemma 1 belongs to $\mathcal{F}(1)$: it is equivalent to the function $f(A) = \gamma \min\{1, |A|\}$.

Two variants of a final example may serve to better understand the flow-cut (and integrality) gap. The first has a large gap, but the rounding methods still find an optimal solution. The second has a gap of one, but the rounding

methods may return solutions with a large approximation factor. Consider a graph with m edges consisting of m/k disjoint paths of length k each (as in Figure 2), with a cost function $f(C) = \max_{e \in C} w(e)$. The edges are partitioned into a cut $B \subset \mathcal{E}$ with $|B| = m/k$ and the remaining edges $\mathcal{E} \setminus B$. Let $w(e) = \gamma$ for $e \notin B$ and $w(e) = \beta$ for $e \in B$.

For the first variant, let $\beta = \gamma$; so that for $k = 1$, we obtain the graph in Lemma 1. With $\beta = \gamma$ (for any k), any minimal cut is optimal, and all rounding methods find an optimal solution. The maximum cooperative flow is $\nu^* = \gamma/k$ (γ/k flow on one path or γ/m flow on each edge in m/k paths in parallel). Hence, the flow-cut gap is $\gamma/(\gamma/k) = k$ despite the optimality of the rounded (and pruned) solutions.

For the second variant, let $\beta = \gamma/k$. The maximum flow remains $\nu^* = \gamma/k$, and the optimal cut is B with $f(B) = \gamma/k$, so $f(C^*) = \nu^*$. An optimal solution y^* to Program (16) is the uniform vector $y^* = (\gamma/m)\mathbf{1}_m$. Despite the zero gap, for such y^* the rounding methods return an arbitrary cut, which can be by a factor k worse than the optimal solution B . In contrast, the approximation algorithms in Sections 5.1.2, 5.1.3 based on substitute cost functions do return an optimal solution.

7 Experiments

We provide a summary of benchmark experiments that compare the proposed algorithms empirically. We use two types of data sets. The first is a collection of average-case submodular cost functions on two types of graph structures, clustered graphs and regular grids. The second consists of a few worst-case examples that show the limits of some of the proposed methods.

The task is to find a minimum cooperative cut in an undirected graph¹¹. This problem can be solved directly or via $n - 1$ minimum (s, t) -cuts. Most of the algorithms solve the (s, t) version. The above approximation bounds still apply, as the minimum cut is the minimum (s, t) -cut for at least one pair of source and sink. We observe that in general, the algorithms perform well, and much better than their theoretical worst-case bounds. Which algorithm is best depends on the cost function and graph at hand.

Algorithms and baselines. Apart from the algorithms discussed in this article, we test some baseline heuristics. First, to test the benefit of the more sophisticated approximations \hat{f}_{ea} and \hat{f}_{pf} we define the simple approximation

$$\hat{f}_{add}(C) = \sum_{e \in C} f(e). \quad (38)$$

The first baseline (MC) simply returns the minimum cut with respect to \hat{f}_{add} . The second baseline (MB) computes the minimum cut basis $\mathcal{C} = \{C_1, \dots, C_{n-1}\}$

¹¹An undirected graph can easily be turned into a directed one by replacing each edge by two opposing directed ones that have the same cost. A cut will always only include one of those edges

with respect to \hat{f}_{add} and then selects $\hat{C} = \operatorname{argmin}_{C \in \mathcal{C}} f(C)$. The minimum cut basis can be computed via a Gomory-Hu tree [11]. As a last baseline, we apply an algorithm by Queyranne [54] to $h(X) = f(\delta(X))$. This algorithm minimizes symmetric submodular functions in $O(n^3)$ time. However, h is only submodular if f is a sum of weights, and therefore this algorithm cannot provide any approximation guarantees here. In fact, we will see in Section 7.2 that it can perform arbitrarily badly.

Of the algorithms described in this article, EA denotes the generic (ellipsoid-based) approximation of Section 5.1.1. The iterative semigradient approximation from Section 5.1.2 is initialized with a random cut basis (RI) or a minimum-weight cut basis (MI). PF is the approximation via polymatroidal network flows (Section 5.1.3). These three approaches approximate the cost functions. In addition, we use algorithms that solve relaxations of Problems (20) and (16): CR solves the convex relaxation using Matlab’s `fmincon`, and applies Algorithm 2 for rounding. DB implements the distance-based rounding by thresholding x^* . Finally, we test the randomized greedy algorithm from Section 5.2.1 with the maximum possible $\beta = \beta_{\max}$ (GM) and an almost maximal $\beta = 0.9\beta_{\max}$ (GA). GH denotes the deterministic greedy heuristic. All algorithms were implemented in Matlab, with the help of a graph cut toolbox [3, 9] and the SFM toolbox [47].

7.1 Average-case

The average-case benchmark data has two components: graphs and cost functions. We first describe the graphs, then the functions.

Grid graphs. The benchmark contains three variants of regular grid graphs of degree four or six. Type I is a plane grid with horizontal and vertical edges displayed as solid edges in Figure 4(a). Type II is similar, but has additional diagonal edges (dashed in Figure 4(a)). Type III is a cube with plane square grids on four faces (sparing the top and bottom faces). Different from Type I, the nodes in the top row are connected to their counterparts on the opposite side of the cube. The connections of the bottom nodes are analogous.

Clustered graphs. The clustered graphs consist of a number of cliques that are connected to each other by few edges, as depicted in Figure 4(b).

Cost functions. The benchmark includes four families of functions. The first group (*Matrix rank I, II, Labels I, II*) consists of matroid rank functions or sums of three such functions. The functions used here are either based on matrix rank or ranks of partition matroids. We summarize those functions as *rank-like* costs.

The second group (*Unstructured I, II*) contains two variants of unstructured functions $g(w(C))$, where g is either a logarithm or a square root. These functions are designed to favor a certain random optimal cut. The construction

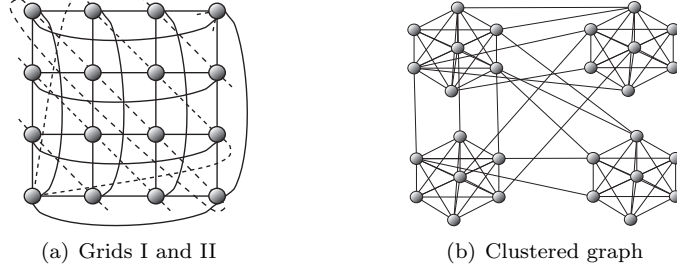


Figure 4: Examples of the test graph structures. The grid (a) was used with and without the dashed diagonal edges, and also with a variation of the connections in the first and last row. The clustered graphs were similar to the example shown in (b).

ensures that the minimum cut will not be one that separates out a single node, but one that cuts several edges.

The third family (*Bestcut I, II*) is constructed to make a cut optimal that has many edges and that is therefore different from the cut that uses fewest edges. For such a cut, we expect \hat{f}_{add} to yield relatively poor solutions.

The fourth set of functions (*Truncated rank*) is inspired by the difficult truncated functions that can be used to establish lower bounds on approximation factors. These functions “hide” an optimal set, and interactions are only visible when guessing a large enough part of this hidden set. The following is a detailed description of all cost functions:

Matrix rank I. Each element $e \in \mathcal{E}$ indexes a column in matrix \mathbf{X} . The cost of $A \subseteq \mathcal{E}$ is the rank of the sub-matrix \mathbf{X}_A of the columns indexed by the $e \in A$: $f_{\text{mrI}}(A) = \text{rank}(\mathbf{X}_A)$. The matrix \mathbf{X} is of the form $[\mathbf{I}' \ \mathbf{R}]$, where $\mathbf{R} \in \{0, 1\}^{d \times (m-d)}$ is a random binary matrix with $d = 0.9\sqrt{m}$, and \mathbf{I}' is a column-wise permutation of the identity matrix.

Matrix rank II. The function $f_{\text{mrII}}(A) = 0.33 \sum_{i=1}^3 f_{\text{mrI}}^{(i)}(A)$ sums up three functions $f_{\text{mrI}}^{(i)}$ of type *matrix rank I* with different random matrices \mathbf{X} .

Labels I. This class consists of functions of the form $f_{\ell\text{I}}(A) = |\bigcup_{e \in A} \ell(e)|$. Each element e is assigned a random label $\ell(e)$ from a set of $0.8\sqrt{m}$ possible labels. The cost counts the number of labels in A .

Labels II. These functions $f_{\ell\text{II}}(A) = 0.33 \sum_{i=1}^3 f_{\ell\text{I}}^{(i)}(A)$ are the sum of three functions of type *labels I* with different random labels.

Unstructured I. These are functions $f_{\text{dPI}}(A) = \log \sum_{e \in A} w(e)$, where weights $w(e)$ are chosen randomly as follows. Sample a set $X \subset V$ with $|X| = 0.4n$, and set $w(e) = 1.001$ for all $e \in \delta X$. Then randomly assign some “heavy” weights in $[n/2, n^2/4]$ to some edges not in δX , so that each node is

incident to one or two heavy edges. The remaining edges get random (mostly integer) weights between 1.001 and $n^2/4 - n + 1$.

Unstructured II. These are functions $f_{\text{dpII}}(A) = \sqrt{\sum_{e \in A} w(e)}$ with weights assigned as for *unstructured function II*.

Bestcut I. We randomly pick a connected subset $X^* \subseteq \mathcal{V}$ of size $0.4n$ and define the cost $f_{\text{bcI}}(A) = \mathbf{1}[|A \cap \delta X^*| \geq 1] + \sum_{e \in A \setminus \delta X^*} w(e)$. The edges in $\mathcal{E} \setminus \delta X^*$ are assigned random weights $w(e) \in [1.5, 2]$. If there is still a cut $C \neq \delta X^*$ with cost one or lower, we correct w by increasing the weight of one $e \in C$ to $w(e) = 2$. The optimal cut is then δX^* , but it is usually not the one with fewest edges.

Bestcut II. Similar to *bestcut I* (δX^* is again optimal), but with submodularity on all edges: \mathcal{E} is partitioned into three sets, $E = (\delta X^*) \cup B \cup C$. Then $f_{\text{bcII}}(A) = \mathbf{1}[|A \cap \delta X^*| \geq 1] + \sum_{e \in A \cap (B \cup C)} w(e) + \max_{e \in A \cap B} w(e) + \max_{e \in A \cap C} w(e)$. The weights of two edges in B and two edges in C are set to $w(e) \in (2.1, 2.2)$.

Truncated rank. This function is similar to the truncated rank in the proof of the lower bound (Theorem 1). Sample a connected $X \subseteq \mathcal{V}$ with $|X| = 0.3|\mathcal{V}|$ and set $R = \delta X$. The cost is $f_{\text{tr}}(A) = \min\{|A \cap \bar{R}| + \min\{|A \cap R|, \lambda_1\}, \lambda_2\}$ for $\lambda_1 = \sqrt{|R|}$ and $\lambda_2 = 2|R|$. Here, R is not necessarily the optimal cut.

To estimate the approximation factor on one problem instance (one graph and one cost function), we divide by the cost of the best solution found by any of the algorithms, unless the optimal solution is known (this is the case for *Bestcut I* and *II*).

7.1.1 Results

Figure 5 shows average empirical approximation factors and also the worst observed factors. The first observation is that all algorithms remain well below their theoretical approximation bounds¹². That means the theoretical bounds are really worst-case results. For several instances we obtain optimal solutions.

The general performance depends much on the actual problem instance; the truncated rank functions with hidden structure are, as may be expected, the most difficult. The simple benchmarks relying on \hat{f}_{add} perform worse than the more sophisticated algorithms. Queyranne's algorithm performs surprisingly well here.

¹²Most of the bounds proved above are absolute, and not asymptotic. The only exception is \hat{f}_{ea} . For simplicity, it is here treated as an absolute bound.

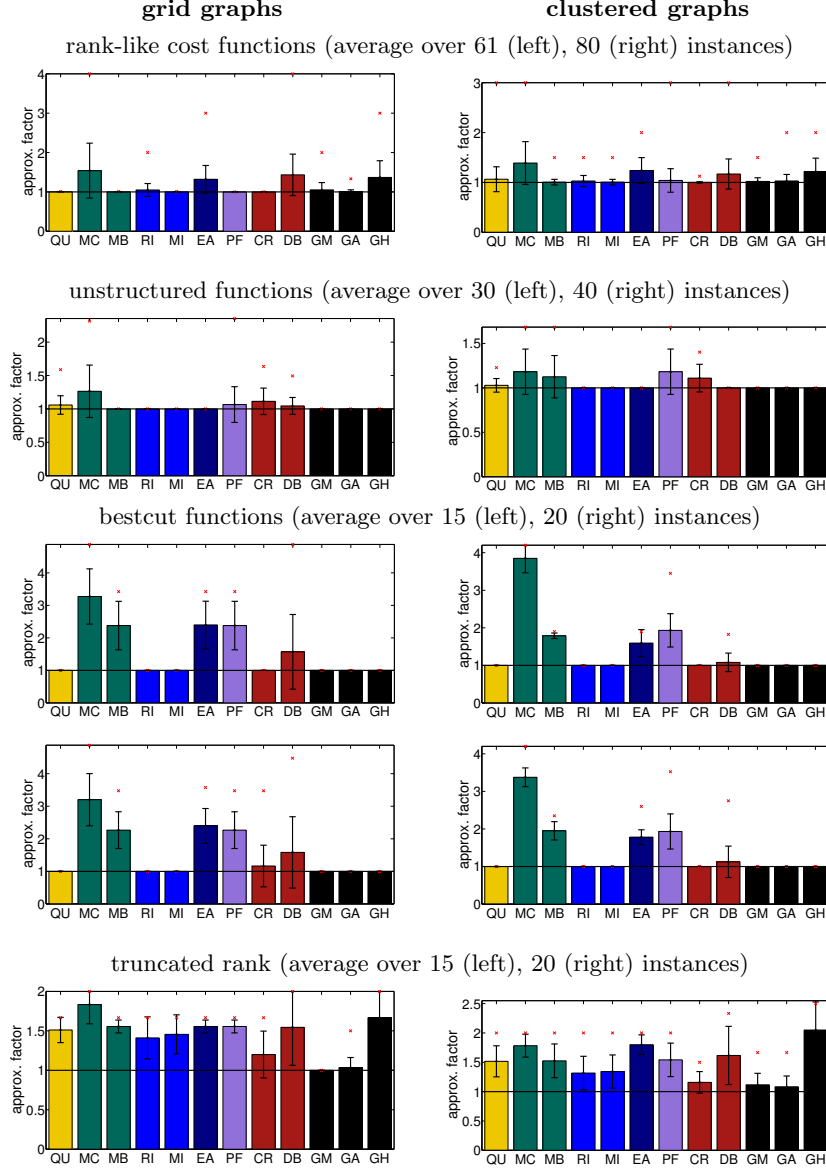


Figure 5: Results for average-case experiments. The bars show the mean empirical approximation factors, and red crosses mark the maximum observed empirical approximation factor. The left column refers to grid graphs, the right column to clustered graphs. The first three algorithms (bars) are baselines, the next four approximate f , the next four solve a relaxation, and the last is the deterministic greedy heuristic.

7.2 Worst-case instances

Lastly, we show two worst-case instances. More examples may be found in [31, Ch. 4]. The example demonstrates the drawbacks of using approximations like \hat{f}_{add} and Queyranne's algorithm.

Our instance is a graph with $n = 10$ nodes, shown in Figure 6. The graph edges are partitioned into $n/2$ sets, indicated by colors. The black set \mathcal{E}_k makes up the cut with the maximum number of edges. The remaining edge sets are constructed as

$$\mathcal{E}_i = \{(v_i, v_j) \in \mathcal{E} \mid i < j \leq n/2\} \cup \{(v_{n/2+i}, v_j) \in \mathcal{E} \mid n/2 + i < j \leq n\} \quad (39)$$

for each $1 \leq i < n/2$. In Figure 6, set \mathcal{E}_1 is red, set \mathcal{E}_2 is blue, and so on. The cost function is

$$f_a(A) = \mathbf{1}[|A \cap \mathcal{E}_k| \geq 1] + \sum_{i=1}^{n/2-1} b \cdot \mathbf{1}[|A \cap \mathcal{E}_i| \geq 1] + \epsilon |A \cap \mathcal{E}_k|, \quad (40)$$

with $b = n/2$. The function $\mathbf{1}[\cdot]$ denotes the indicator function. The cost of the optimal solution is $f(C^*) = f(\mathcal{E}_k) = 1 + \frac{n^2}{4}\epsilon \approx 1$. The second-best solution is the cut $\delta(v_1)$ with cost $f(\delta v_1) = 1 + \frac{n^2}{4}\epsilon + b \approx 1 + \frac{n}{2} = 6$, i.e., it is by a factor of almost $b = n/2$ worse than the optimal solution. Finally, MC finds the solution $\delta(v_n)$ with $f(\delta v_n) = 1 + \frac{n^2}{4}\epsilon + b(\frac{n}{2} - 1) \approx \frac{n^2}{4} = 21$.

Variant (b) uses the cost function

$$f_b(A) = \mathbf{1}[|A \cap \mathcal{E}_k| \geq 1] + \sum_{i=1}^{n/2-1} b \cdot \mathbf{1}[|A \cap \mathcal{E}_i| \geq 1] \quad (41)$$

with a large constant $b = n^2 = 100$. For any $b > n/2$, any solution other than C^* is more than $n^2/4 = |C^*| > n$ times worse than the optimal solution. Hence, thanks to the upper bounds on their approximation factors, all algorithms except for QU find the optimal solution. The result of the latter depends on how it selects a minimizer of $f(B \cup e) - f(e)$ in the search for a pendent pair; this quantity often has several minimizers here. Variant (b) uses a specific adversarial permutation of node labels, for which QU always returns the same solution δv_1 with cost $b + 1$, no matter how large b is: its solution can become arbitrarily poor.

8 Discussion

In this work, we have analyzed the MINCOOPCUT problem, that is, a minimum (s, t) -cut problem with a submodular cost function on graph edges. This problem unifies a number of nonlinear graph cut problems in the literature that have arisen in different application areas.

We showed an information-theoretic lower bound of $\Omega(\sqrt{n})$ for the general MINCOOPCUT problem if the function is given as an oracle, and NP-hardness

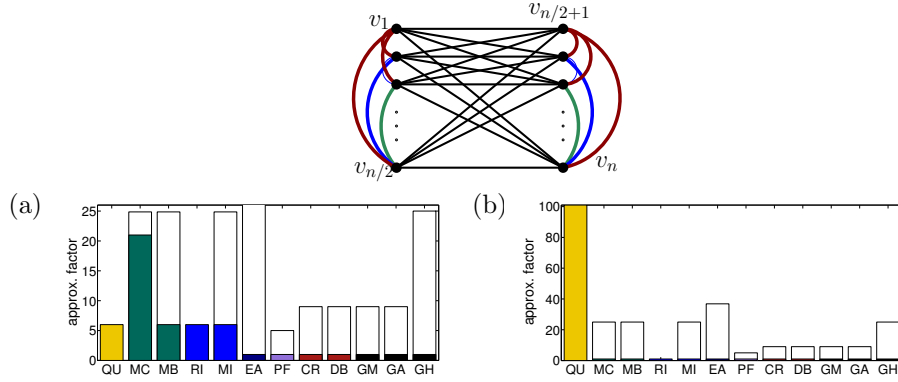


Figure 6: Worst-case instance and empirical approximation factors with $n = 10$ nodes. White bars illustrate theoretical approximation bounds where applicable. In (b), the second-best cut δv_1 has cost $f_b(\delta v_1) = b + 1 = 101 \gg \max\{|C^*|, n, \sqrt{m} \log m\}$.

even if the cost function is fully known and polynomially representable. We propose and compare complementary approximation algorithms that either rely on representing the cost function by a simpler function, or on solving a relaxation of the mathematical program. The latter are closely tied to the longest path of cooperating edges in the graph, as is the flow-cut gap. We also show that the flow-cut gap may be as large as $n - 1$, and therefore larger than the best approximation factor possible.

The lower bound and analysis of the integrality gap use a particular graph structure, a graph with parallel disjoint paths of equal length. Taken all proposed algorithms together, all instances of MINCOOPCUT on graphs with parallel paths of the same length can be solved within an approximation bound at most \sqrt{n} . This leaves the question whether there is an instance that makes *all* approximations worse than \sqrt{n} .

Section 6 outlined properties of submodular functions that facilitate submodular minimization under combinatorial constraints, and also submodular minimization in general. Apart from separability, we defined the hierarchy of function classes $\mathcal{F}(k)$. The $\mathcal{F}(k)$ are related to graph-representability and might therefore build a bridge between recent results about limitations of representing submodular functions as graph cuts [67] (and, even stricter, the limitations of polynomial representability) and the results discussed in Section 6.2 that provide improved algorithms whose complexity depends on k .

8.1 Cooperative Multi-cut and Sparsest cut

An extension of MINCOOPCUT is the problem of cooperative multi-way cut and sparsest cut. Using the approximation \hat{f}_{ea} from Section 5.1.3, we can transform any multi-way or sparsest cut problem with a submodular cost function on

edges (instead of a sum of edge weights) into a cut problem whose cut cost is a convolution of local submodular functions. The relaxation of this cut problem is dual to the polymatroidal flow problems considered by Chekuri et al. [13]. Combining their results with ours, we get the following Lemma.

Lemma 11. *Let α be the approximation factor for solving a sparsest cut / multi-way cut in a polymatroidal network. If we solve a cooperative sparsest cut / multi-way cut by first approximating the cut cost f by a function \hat{f}_{ea} and, on this instance, using the method with factor α , we get an $O(\alpha n)$ -approximation for cooperative sparsest cut / multi-way cut.*

Using Theorems 6 and 8 in [13], we obtain for example the following bounds:

Corollary 2. *There is an $O(n \log k)$ approximation for cooperative sparsest cut in undirected graphs that is dual to a maximum multicommodity flow problem with k pairs, and an $O(n \log k)$ approximation for cooperative multi-way cut.*

We leave it as an open problem whether these bounds are optimal.

Acknowledgments

The authors would like to thank Chandra Chekuri for suggesting the rounding method in Lemma 10.

References

- [1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice Hall, 1993.
- [2] C. Allène, J.-Y. Audibert, M. Couprie, and R. Keriven. Some links between extremum spanning forests, watersheds, and min-cuts. *Image and Vision Computing*, 2009.
- [3] S. Bagon. Matlab wrapper for graph cut, December 2006. <http://www.wisdom.weizmann.ac.il/~bagon>.
- [4] N. Balcan and N. Harvey. Submodular functions: Learnability, structure, and optimization. *arXiv:0486478*, 2012.
- [5] F. Baumann, S. Berckey, and C. Buchheim. *Facets of Combinatorial Optimization — Festschrift for Martin Grötschel*, chapter Exact Algorithms for Combinatorial Optimization Problems with Submodular Objective Functions, pages 271–294. Springer, 2013.
- [6] J. Bilmes. Dynamic graphical models – an overview. *IEEE Signal Processing Magazine*, 27(6):29–42, 2010.
- [7] J. Bilmes and C. Bartels. On triangulating dynamic graphical models. In *Uncertainty in Artificial Intelligence*, pages 47–56, Acapulco, Mexico, 2003. Morgan Kaufmann Publishers.

- [8] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Int. Conf. on Computer Vision (ICCV)*, 2001.
- [9] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.
- [10] Y. Boykov and O. Veksler. *Handbook of Mathematical Models in Computer Vision*, chapter Graph Cuts in Vision and Graphics: Theories and Applications. Springer, 2006.
- [11] F. Bunke, H. W. Hamacher, F. Maffioli, and A. Schwahn. Minimum cut bases in undirected networks. Report in Wirtschaftsmathematik (WIMA Report) 108, Universität Kaiserslautern, 2007.
- [12] A. Chambolle and J. Darbon. On total variation minimization and surface evolution using parametric maximum flows. *Int. Journal of Computer Vision*, 84(3), 2009.
- [13] C. Chekuri, S. Kannan, A. Raja, and P. Viswanath. Multicommodity flows and cuts in polymatroidal networks. In *Innovations in Theoretical Computer Science (ITCS)*, 2012.
- [14] M. Conforti and G. Cornuéjols. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete Applied Mathematics*, 7(3):251–274, 1984.
- [15] C. Couprie, L. Grady, H. Talbot, and L. Najman. Combinatorial continuous maximum flow. *SIAM Journal on Imaging*, pages 905–930, 2011.
- [16] W. H. Cunningham. Decomposition of submodular functions. *Combinatorica*, 3(1):53–68, 1983.
- [17] G.B. Dantzig and D.R. Fulkerson. On the max flow min cut theorem of networks. Technical Report P-826, The RAND Corporation, 1955.
- [18] L.R. Ford and D.R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.
- [19] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.
- [20] G. Goel, C. Karande, P. Tripathi, and L. Wang. Approximability of combinatorial problems with multi-agent submodular cost functions. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS)*, 2009.

- [21] G. Goel, P. Tripathi, and L. Wang. Combinatorial problems with discounted price functions in multi-agent systems. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2010.
- [22] M. X. Goemans, N. J. A. Harvey, R. Kleinberg, and V. S. Mirrokni. On learning submodular functions – a preliminary draft. Unpublished Manuscript.
- [23] M.X. Goemans, N. J. A. Harvey, S. Iwata, and V. S. Mirrokni. Approximating submodular functions everywhere. In *Proc. SIAM-ACM Symp. on Discrete Algorithms (SODA)*, 2009.
- [24] V. Goyal and R. Ravi. An FPTAS for minimizing a class of low-rank quasiconcave functions over a convex domain. Technical Report 366, Tepper School of Business, Carnegie Mellon University, 2008.
- [25] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society*, 51(2), 1989.
- [26] R. Hassin. Minimum cost flow with set constraints. *Networks*, 12:1–21, 1982.
- [27] R. Hassin, J. Monnot, and D. Segev. Approximation algorithms and hardness results for labeled connectivity problems. *J. Comb. Optim.*, 14(4): 437–453, 2007.
- [28] S. Iwata and K. Nagano. Submodular function minimization under covering constraints. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS)*, 2009.
- [29] R. Iyer, S. Jegelka, and J. Bilmes. Fast semidifferential-based submodular function optimization. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2013.
- [30] R. Iyer, S. Jegelka, and J. Bilmes. Curvature and optimal algorithms for learning and minimizing submodular functions. In *Neural Information Processing Society (NIPS)*, 2013.
- [31] S. Jegelka. *Combinatorial Problems with submodular coupling in machine learning and computer vision*. PhD thesis, ETH Zurich, 2012.
- [32] S. Jegelka and J. Bilmes. Submodularity beyond submodular energies: coupling edges in graph cuts. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [33] S. Jegelka and J. Bilmes. Approximation bounds for inference using cooperative cuts. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2011.
- [34] S. Jegelka, H. Lin, and J. Bilmes. On fast approximate submodular minimization. In *Neural Information Processing Society (NIPS)*, 2011.

- [35] S. Jegelka, F. Bach, and S. Sra. Reflection methods for user-friendly submodular optimization. In *Neural Information Processing Society (NIPS)*, 2013.
- [36] S. Jha, O. Sheyner, and J.M. Wing. Two formal analyses of attack graphs. In *Proc. of the 15th Computer Security Foundations Workshop*, pages 49–63, 2002.
- [37] S. Kannan and P. Viswanath. Multiple-unicast in fading wireless networks: A separation scheme is approximately optimal. In *IEEE Int. Symposium on Information Theory (ISIT)*, 2011.
- [38] S. Kannan, A. Raja, and P. Viswanath. Local phy + global flow: A layering principle for wireless networks. In *IEEE Int. Symposium on Information Theory (ISIT)*, 2011.
- [39] P. Kohli, M. P. Kumar, and P. Torr. P³ & beyond: solving energies with higher-order cliques. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007.
- [40] P. Kohli, M.P. Kumar, and P.H.S. Torr. P³ & beyond: Move making algorithms for solving higher order functions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 1645–1656, 2009.
- [41] P. Kohli, L. Ladický, and P.H.S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.
- [42] P. Kohli, A. Osokin, and S. Jegelka. A principled deep random field for image segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013.
- [43] V. Kolmogorov. Minimizing a sum of submodular functions. *Discrete Applied Mathematics*, 160(15), 2012.
- [44] V. Kolmogorov and Y. Boykov. What metrics can be approximated by geo-cuts, or global optimization of length/area and flux. In *Int. Conf. on Computer Vision (ICCV)*, 2005.
- [45] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.
- [46] C. Koufogiannakis and N. E. Young. Greedy Δ -approximation algorithm for covering with arbitrary constraints and submodular costs. In *Int. Colloquium on Automata, Languages and Programming (ICALP)*, 2009.
- [47] A. Krause. Matlab toolbox for submodular function optimization, 2009. <http://www.cs.caltech.edu/~krausea/sfo/>.

- [48] E. L. Lawler and C. U. Martel. Computing maximal “Polymatroidal” network flows. *Mathematics of Operations Research*, 7(3):334–347, 1982.
- [49] L. Lovász. *Mathematical programming – The State of the Art*, chapter Submodular Functions and Convexity, pages 235–257. Springer, 1983.
- [50] S. Mittal and A. Schulz. An FPTAS for optimizing a class of low-rank functions over a polytope. *Mathematical Programming*, 2012.
- [51] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.
- [52] H. Narayanan. *Submodular Functions and Electrical Networks*. Elsevier Science, 1997.
- [53] E. Nikolova. Approximation algorithms for reliable stochastic combinatorial optimization. In *APPROX*, 2010.
- [54] M. Queyranne. Minimizing symmetric submodular functions. *Mathematical Programming*, 82:3–12, 1998.
- [55] S. Ramalingam, P. Kohli, K. Alahari, and P. Torr. Exact inference in multi-label crfs with higher order cliques. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.
- [56] S. Ramalingam, C. Russell, L. Ladicky, and P. H. S. Torr. Efficient minimization of higher order submodular functions using monotonic boolean functions. *ArXiv 1109.2304*, 2011.
- [57] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, (60), 1992.
- [58] A. Schrijver. *Combinatorial Optimization*. Springer, 2004.
- [59] A.K. Sinop and L. Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *Int. Conf. on Computer Vision (ICCV)*, 2007.
- [60] R. P. Stanley. *Enumerative Combinatorics*, volume I of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1997.
- [61] P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *Neural Information Processing Society (NIPS)*, 2010.
- [62] Z. Svitkina and L. Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS)*, 2008.
- [63] E. Tardos, C. A. Tovey, and M. A. Trick. Layered augmenting path algorithms. *Mathematics of Operations Research*, 11(2), 1986.

- [64] J. Vondrák. Submodularity and curvature: the optimal algorithm. *RIMS Kôkyûroku Bessatsu*, 2008.
- [65] J. Vondrák. Symmetry and approximability of submodular maximization problems. Technical Report arXiv:1110.4860v1, 2011.
- [66] P. Zhang, Cai J.-Y, L.-Q. Tang, and W.-B. Zhao. Approximation and hardness results for label cut and related problems. *Journal of Combinatorial Optimization*, 2011.
- [67] S. Živný, D. A. Cohen, and P. G. Jeavons. The expressive power of binary submodular functions. *Discrete Applied Mathematics*, 157(15):3347–3358, 2009.

A Proof of Proposition 1

The first part of Proposition 1 is proven by Figure 1. Here, we show the second part that the function $h(X) = f(\delta^+(X))$ is subadditive if f is nondecreasing and submodular. Let $X, Y \subseteq \mathcal{V}$. Then it holds that

$$h(X) + h(Y) = f(\delta^+(X)) + f(\delta^+(Y)) \quad (42)$$

$$\geq f(\delta^+(X) \cup \delta^+(Y)) + f(\delta^+(X) \cap \delta^+(Y)) \quad (43)$$

$$\geq f(\delta^+(X) \cup \delta^+(Y)) \quad (44)$$

$$\geq f(\delta^+(X \cup Y)) \quad (45)$$

$$= h(X \cup Y). \quad (46)$$

In Inequality (43), we used that f is submodular, and in Inequality (44), we used that f is nonnegative.

B Reduction from GRAPH BISECTION to MINCOOP-CUT

In this section, we prove Theorem 2 via a reduction from GRAPH BISECTION, which is known to be NP-hard [19].

Definition 2 (GRAPH BISECTION). *Given a graph $\mathcal{G}_B = (\mathcal{V}_B, \mathcal{E}_B)$ with edge weights $w_B : \mathcal{E}_B \rightarrow \mathbb{R}_+$, find a partition $V_1 \dot{\cup} V_2 = \mathcal{V}_B$ with $|V_1| = |V_2| = |\mathcal{V}_B|/2$ with minimum cut weight $w(\delta(V_1))$.*

Proof. To reduce GRAPH BISECTION to MINCOOPCUT, we construct an auxiliary graph with two additional terminal nodes. The submodular edge weights on the edges adjacent to the terminal nodes will express the balance constraint $|V_1| = |V_2| = |\mathcal{V}_B|/2$. The edge weights on the instance for graph bisection remain unchanged. The proof involves three graphs: a given instance \mathcal{G}_B of

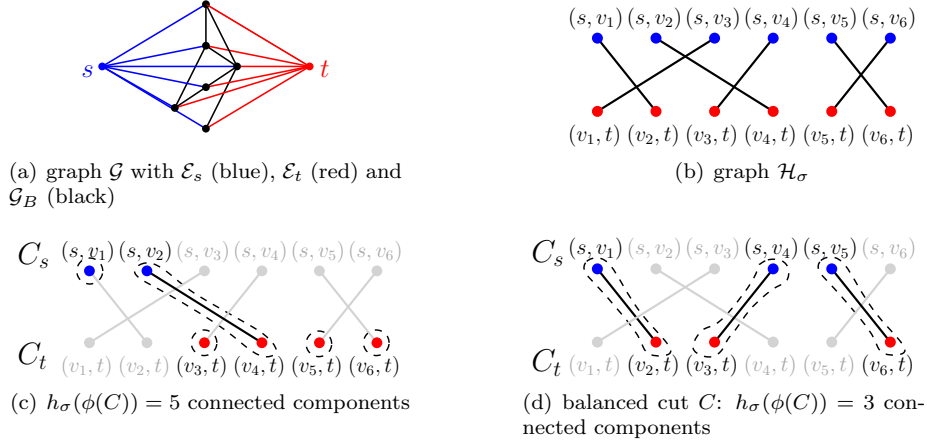


Figure 7: Graph for the reduction and examples for the definition of f_{bal} via ranks h_σ , with $n_B = 6$. In (c), $C_s = \{(s, v_1), (s, v_2)\}$ and $C_t = \{(v_3, t), (v_4, t), (v_5, t), (v_6, t)\}$; in (d), $C_s = \{(s, v_1), (s, v_4), (s, v_5)\}$ and $C_t = \{(v_2, t), (v_3, t), (v_6, t)\}$. Connected components are indicated by dashed lines.

GRAPH BISECTION, a graph \mathcal{G} that has a cooperative cut cost function and represents the graph bisection instance $\mathcal{G}_B = (\mathcal{V}_B, \mathcal{E}_B)$, and a graph \mathcal{H}_σ that defines the cost function on \mathcal{G} .

To form \mathcal{G} , we retain \mathcal{G}_B with the modular costs on \mathcal{E}_B , add nodes s, t and connect those to every vertex in \mathcal{G}_B with corresponding new edge sets \mathcal{E}_s and \mathcal{E}_t . That means, $\mathcal{G} = (\mathcal{V}_B \cup \{s, t\}, \mathcal{E}_B \cup \mathcal{E}_s \cup \mathcal{E}_t)$, as Figure 7(a) shows. The cost of a cut in the auxiliary graph \mathcal{G} is measured by the submodular function

$$f(C) = \sum_{e \in C \cap \mathcal{E}_B} w(e) + \beta f_{bal}(C \cap (\mathcal{E}_s \cup \mathcal{E}_t)), \quad (47)$$

where β is an appropriately large constant, and f_{bal} will be defined later. The cost f_{bal} on $\mathcal{E}_s \cup \mathcal{E}_t$ implements the equipartition constraint on \mathcal{V}_B . Obviously, any (s, t) -cut C must include at least $n_B = |\mathcal{V}_B|$ edges from $\mathcal{E}_s \cup \mathcal{E}_t$. A minimal cut assigns $v \in \mathcal{V}_B$ to t by cutting (s, v) , and to s by cutting (v, t) , and thus defines a partition of \mathcal{V}_B . As a result, the cardinality of $C_s = C \cap \mathcal{E}_s$ is the number of nodes in \mathcal{V}_B assigned to t . An analogous equivalence holds for $C_t = C \cap \mathcal{E}_t$. In an equipartition, $|C_s| = |C_t| = n_B/2$.

We now implement the equipartition constraint by a submodular, nondecreasing cost on $\mathcal{E}_s \cup \mathcal{E}_t$. The function will be a sum of matroid rank functions h_σ . Each h_σ is based on a bipartite graph $\mathcal{H}_\sigma = (\mathcal{E}_s, \mathcal{E}_t, \mathcal{F}_\sigma)$ that has nodes $\mathcal{E}_s \cup \mathcal{E}_t$. Its edges \mathcal{F}_σ form a derangement¹³ σ between nodes from \mathcal{E}_s and \mathcal{E}_t , as illustrated in Fig. 7(b). We denote by $\phi(C_s \cup C_t)$ the image of $C_s \cup C_t$ in the set of nodes of \mathcal{H}_σ . Let the rank function $h_\sigma : 2^{\phi(\mathcal{E}_s \cup \mathcal{E}_t)} \rightarrow \mathbb{N}_0$ count the number

¹³A *derangement* is a permutation that maps no element to itself.

of connected components in the subgraph induced by the nodes $\phi(C_s \cup C_t)$. Figure 7 shows some examples. Each derangement on n_B items induces such a rank¹⁴.

Let \mathfrak{S} be the set of all derangements σ of n_B elements, i.e., all possible edge configurations in the graphs \mathcal{H}_σ . We define f_{bal} to be the expectation of h_σ if $\sigma \in \mathfrak{S}$ is drawn uniformly at random:

$$f_{bal}(C) = \mathbb{E}_\sigma[h_\sigma(\phi(C))] = |\mathfrak{S}|^{-1} \sum_{\sigma \in \mathfrak{S}} h_\sigma(\phi(C)). \quad (48)$$

For a fixed derangement σ' and a fixed size $|C_s \cup C_t| = n_B$, the value $h_{\sigma'}(C_s \cup C_t)$ is minimal if the number of matched nodes is maximal. Then $\sigma'(C_s) = C_t$ and $|C_s| = |C_t|$.

To compute the rank $h_\sigma(C)$ for a fixed σ , we sum up all nodes $\phi(C_s \cup C_t) = |C_s| + |C_t|$. Then we subtract the number of matches, because those components were counted twice. To shorten notation, we denote the node (s, v_i) in \mathcal{H}_σ by x_i , and its counterpart (v_i, t) by y_i . Formally, the rank is

$$h_\sigma(\phi(C_s) \cup \phi(C_t)) = |C_s| + |C_t| - \left| \{(x_i, y_{\sigma(i)})\}_{i=1}^n \cap (\phi(C_s) \times \phi(C_t)) \right|. \quad (49)$$

As an average of rank functions, f_{bal} is submodular and monotone. From Equation (49) it follows that the sum (48) consists of two terms:

$$\sum_{\sigma \in \mathfrak{S}} h_\sigma(C) = |\mathfrak{S}|(|C_s| + |C_t|) - \sum_{\sigma \in \mathfrak{S}} \left| \{(x_i, y_{\sigma(i)})\}_{i=1}^n \cap (\phi(C_s) \times \phi(C_t)) \right| \quad (50)$$

$$= |\mathfrak{S}|(|C_s| + |C_t|) - \sum_{x_i \in \phi(C_s)} \sum_{\sigma \in \mathfrak{S}} \left| (x_i, y_{\sigma(i)}) \cap (\{x_i\} \times \phi(C_t)) \right| \quad (51)$$

That means we count the total number of matches as the sum of the number of matches for each x_i in $\phi(C_s)$. To count the matches of a fixed $x_i \in \phi(C_s)$, we calculate how many derangements map it to an element in $\phi(C_t)$ and yield a match.

When counting, we must regard that σ is a derangement, so there will never be an edge (x_i, y_i) in \mathcal{H}_σ . Let $C_{s \cap t} \triangleq \{(s, v) \mid \{(s, v), (v, t)\} \subseteq C\}$ be the set of s -edges whose counterpart on the t side is also contained in C . This set is nonempty if C cuts off a node from both s and t . Each element x_i in $\phi(C_s \setminus C_{s \cap t})$ can be mapped by σ to any element $y_k \in \phi(C_t)$. For each such (fixed) pairing (x_i, y_k) , any of the remaining $n_B - 1$ elements x_j can be mapped to any y_ℓ with $j \neq \ell$. Moreover, the element x_k can be mapped to any remaining target in \mathcal{E}_t , since its counterpart y_k is already “taken” by x_i . Let $D'(n_B - 1)$ denote the number of permutations of $n_B - 1$ elements (pair (x_i, y_k) , i.e., $\sigma(i) = k$, is fixed), where one specific element x_k can be mapped to any other of the $n_B - 1$ elements, and the remaining elements must not be mapped to their counterparts ($\sigma(j) \neq j$). Then there are $D'(n_B - 1)$ derangements σ realizing $\sigma(i) = k$, for

¹⁴This function is the rank of a partition matroid. Clearly, the edges in each derangement partition the set of nodes into sets of size 2.

each $y_k \in \phi(C_t)$. This makes $|C_t|D'(n_B - 1)$ matches for each x_i in $\phi(C_s \setminus C_{s \cap t})$, and so we count $|C_s \setminus C_{s \cap t}||C_t|D'(n_B - 1)$ matches in total for the $x_i \in C_s \setminus C_{s \cap t}$.

Each element x_i in the remaining $\phi(C_{s \cap t})$ can be mapped to $|C_t| - 1$ elements in $\phi(C_t)$, since its counterpart y_i is in $\phi(C_t)$. With a similar count as above, this leads to another $|C_{s \cap t}|(|C_t| - 1)D'(n_B - 1)$ matches. Let $D(n)$ be the number of derangements of n elements. In total, we get

$$D(n_B)f_{bal}(C) = (|C_s| + |C_t|)D(n_B) \quad (52)$$

$$- \sum_{x_i \in C_s \setminus C_{s \cap t}} \sum_{y_k \in C_t} D'(n_B - 1) - \sum_{x_i \in C_{s \cap t}} \sum_{y_k \in C_t, k \neq i} D'(n_B - 1)$$

$$= (|C_s| + |C_t|)D(n_B) \quad (53)$$

$$- (|C_s| - |C_{s \cap t}|)|C_t|D'(n_B - 1) - |C_{s \cap t}|(|C_t| - 1)D'(n_B - 1)$$

$$= (|C_s| + |C_t|)D(n_B) - (|C_s||C_t| - |C_{s \cap t}|)D'(n_B - 1), \quad (54)$$

with $D(n) = |\mathfrak{S}| = n! \sum_{k=0}^n (-1)^k / k!$ [60], and $D'(n - 1) = \sum_{k=0}^{n-1} (n - 2)!(n - 1 - k)!(-1)^k$ (derived in Section B.1). The derangements lead to the penalty $|C_{s \cap t}|$ for overlaps.

Given that $|C_s| + |C_t|$ must cut at least n_B edges and that f_{bal} is increasing, f_{bal} is minimized if $|C_s| = |C_t| = n_B/2$. In that case, $n_B/2$ nodes are assigned to s and $n_B/2$ to t . As a result, if β is large enough such that f_{bal} dominates the cost, then a minimum cooperative cut in \mathcal{G} bisects the \mathcal{G}_B subgraph of \mathcal{G} optimally. \square

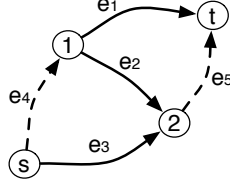
B.1 Derivation of $D'(n)$

In this section, we derive the number $D'(n)$ of modified derangements that was used to prove MINCOOPCUT to be NP-hard. A derangement is a permutation, i.e., a mapping $\sigma : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$, where no element can be mapped to itself: $\sigma(i) \neq i$ for all $1 \leq i \leq n$. We define a relaxed version of a derangement, where one pre-specified element i' can be mapped to itself, but no other element can: $\sigma(i') \in \{1, \dots, n\}$, but $\sigma(i) \neq i$ for all $i \neq i'$. The number $D'(n)$ is the number of such relaxed derangements given a specific i' .

We derive $D'(n)$ by the method of the forbidden board [60, pp. 71-73]. Let, without loss of generality, $i' = n$. Then the forbidden board is $B = \{(1, 1), (2, 2), \dots, (n - 1, n - 1)\}$. Let N_j be the number of permutations σ for which $|\{(i, \sigma(i))\}_{i=1}^n \cap B| = j$; the graph of these permutations coincides with B in j positions. Furthermore, let r_k be the number of k -subsets of B such that no two elements have a coordinate in common. The polynomial

$$N_n(x) = \sum_j N_j x^j = \sum_{k=0}^n r_k (n - k)!(x - 1)^k \quad (55)$$

gives the desired solution $D'(n) = N_0 = N_n(0)$. For the board B above,



Let $f(A) = \max_{e \in A} w(e)$ and
 $w(e_1) = w(e_2) = a$,
 $w(e_3) = b$,
 $w(e_4) = w(e_5) = \epsilon$.

Figure 8: Example showing that the convolution of submodular functions is not always submodular, e.g., for $a = 1.5$, $b = 2$ and $\epsilon = 0.001$.

$r_k = \binom{n-1}{k}$. Thus,

$$N_n(x) = \sum_{k=0}^n r_k (n-k)! (x-1)^k \quad (56)$$

$$= \sum_{k=0}^n \binom{n-1}{k} (n-k)! (x-1)^k \quad (57)$$

$$= \sum_{k=0}^n \frac{(n-1)!}{k!(n-1-k)!} (n-k)! (x-1)^k \quad (58)$$

$$= \sum_{k=0}^n \frac{(n-1)!}{k!} (n-k) (x-1)^k. \quad (59)$$

Then $D'(n) = N_n(0) = \sum_{k=0}^n \frac{(n-1)!}{k!} (n-k)! (-1)^k$ and $D'(n-1) = N_{n-1}(0) = \sum_{k=0}^{n-1} \frac{(n-2)!}{k!} (n-1-k)! (-1)^k$.

C Convolutions of submodular functions are not always submodular

The non-submodularity of convolutions was mentioned already in [49]. For completeness, we show an explicit example that illustrates that non-submodularity also holds for the special case of polymatroidal flows.

Proposition 2. *The convolution of two submodular functions $(f * g)(A) = \min_{B \subseteq A} f(B) + g(A \setminus B)$ is not in general submodular. In particular, this also holds for the cut cost functions occurring in the dual problems of polymatroidal maximum flows.*

To show Proposition 2, consider the graph in Figure 2 with a submodular edge cost function $f(A) = \max_{e \in A} w(e)$. The two submodular functions that

are convolved in the corresponding polymatroidal flow are the decompositions

$$\text{cap}^{\text{out}}(A) = \sum_{v \in \mathcal{V}} f(A \cap \delta^+(v)) \quad (60)$$

$$\text{cap}^{\text{in}}(A) = \sum_{v \in \mathcal{V}} f(A \cap \delta^-(v)). \quad (61)$$

Note that both functions by themselves are a submodular function from $2^{\mathcal{E}}$ to \mathbb{R}_+ . Their convolution is the function h defined as

$$h(A) = (\text{cap}^{\text{out}} * \text{cap}^{\text{in}})(A) = \min_{B \subseteq A} \text{cap}^{\text{out}}(B) + \text{cap}^{\text{in}}(A \setminus B) = \hat{f}_{pf}(A). \quad (62)$$

For h to be submodular, it must satisfy the condition of diminishing marginal costs, i.e., for any e and $A \subseteq B \subseteq \mathcal{E} \setminus e$, it must hold that $h(e \mid A) \geq h(e \mid B)$. Now, let $A = \{e_2\}$ and $B = \{e_1, e_2\}$. The convolution here basically means to pair e_2 either with e_1 or e_2 . Then, if $a < b$,

$$h(e_3 \mid A) = \min\{a + b, b\} - a = b - a \quad (63)$$

$$h(e_3 \mid B) = a + b - \min\{a + a, a\} = b. \quad (64)$$

Hence, $h(e_3 \mid A) < h(e_3 \mid B)$, disproving submodularity of h .

D Cooperative Cuts and Polymatroidal Networks

First, we prove Lemma 6 that relates the approximation \hat{f}_{pf} to maxflow problems in polymatroidal networks.

Proof. (Lemma 6) First, we state the dual of a polymatroidal flow. Let $\text{cap}^{\text{in}} : 2^{\mathcal{E}} \rightarrow \mathbb{R}_+$ be the joint incoming capacity, $\text{cap}^{\text{in}}(C) = \sum_{v \in V} \text{cap}_v^{\text{in}}(C \cap \delta^-v)$, and let equivalently cap^{out} be the joint outgoing capacity. The dual of the polymatroidal maximum flow is a minimum cut problem whose cost is a convolution of edge capacities [49]:

$$\text{cap}(C) = (\text{cap}^{\text{in}} * \text{cap}^{\text{out}})(C) \triangleq \min_{A \subseteq C} [\text{cap}^{\text{in}}(A) + \text{cap}^{\text{out}}(C \setminus A)]. \quad (65)$$

We will relate this dual to the approximation \hat{f}_{pf} . Given a minimal (s, t) -cut C , let $\Pi(C)$ be a partition of C , and $C_v^{\text{in}} = C_v^{\Pi} \cap \delta_v^-$ and $C_v^{\text{out}} = C_v^{\Pi} \cap \delta_v^+$. The cut C partitions the nodes into two sets \mathcal{V}_s containing s and \mathcal{V}_t containing t . Since C is a minimal directed cut, it contains only edges from the s side \mathcal{V}_s to the t side \mathcal{V}_t of the graph. In consequence, $C_v^{\text{in}} = \emptyset$ if v is on the s side, and $C_v^{\text{out}} = \emptyset$ otherwise. Hence, $C_v^{\text{in}} \cup C_v^{\text{out}}$ is equal to either C_v^{in} or C_v^{out} , and since $f(\emptyset) = 0$, it holds that $f(C_v^{\text{in}} \cup C_v^{\text{out}}) = f(C_v^{\text{in}}) + f(C_v^{\text{out}})$. Then, starting with

the definition of \hat{f}_{pf} ,

$$\hat{f}_{pf}(C) = \min_{\Pi(C) \in \mathcal{P}_C} \sum_{v \in \mathcal{V}} f(C_v^\Pi) \quad (66)$$

$$= \min_{\Pi(C) \in \mathcal{P}_C} \sum_{v \in \mathcal{V}} f(C_v^{\text{in}} \cup C_v^{\text{out}}) \quad (67)$$

$$= \min_{\Pi(C) \in \mathcal{P}_C} \sum_{v \in \mathcal{V}} [f(C_v^{\text{in}}) + f(C_v^{\text{out}})] \quad (68)$$

$$= \min_{\Pi(C) \in \mathcal{P}_C} \sum_{v \in \mathcal{V}} [\text{cap}_v^{\text{in}}(C_v^{\text{in}}) + \text{cap}_v^{\text{out}}(C_v^{\text{out}})] \quad (69)$$

$$= \min_{C^{\text{in}}, C^{\text{out}}} [\text{cap}^{\text{in}}(C^{\text{in}}) + \text{cap}^{\text{out}}(C^{\text{out}})] \quad (70)$$

$$= \min_{C^{\text{in}} \subseteq C} [\text{cap}^{\text{in}}(C^{\text{in}}) + \text{cap}^{\text{out}}(C \setminus C^{\text{in}})] \quad (71)$$

$$= (\text{cap}^{\text{in}} * \text{cap}^{\text{out}})(C). \quad (72)$$

The minimum in Equation (68) is taken over all feasible partitions $\Pi(C)$ and their resulting intersections with the sets δ^+v, δ^-v . Then we use the notation $C^{\text{in}} = \bigcup_{v \in \mathcal{V}} C_v^{\text{in}}$ for all edges assigned to their head nodes, and $C^{\text{out}} = \bigcup_{v \in \mathcal{V}} C_v^{\text{out}}$. The minima in Equations (70) and (71) are again taken over all partitions in \mathcal{P}_C . The final equality follows from the above definition of a convolution of submodular functions. \square \square